A Layered UVM Based Testbench Design for SpaceWire

Ahmet Cagri Bagbaba[1], Buse Ustaoglu[1], Inan Erdem[2], Berna Ors[1]

Istanbul Technical University[1], Anka Microelectronic Systems[2]

{bagbaba,ustaoglubu,Siddika.Ors}@itu.edu.tr,inan.erdem@ankasys.com

Introduction

In this work, a layered UVM based verification IP is designed with an emphasis of making the analysis and reporting easier for the verification of SpaceWire based designs. This implementation can also be used as the SpaceWire VIP to help further analyze the SpaceWire networks by looking into the data at different SpaceWire layers in. The implementation is emitting transactions at all SpaceWire protocol layers to the user. At the signal layer, the VIP emits the bits that it captures by decoding the DS-encoded Spacewire signals. The VIP itself uses these transactions internally to capture the disconnect error conditions and forwards them to the character encoder component. The character encoder then takes the incoming bit streams and extracts the SpaceWire characters and codes out of it. After having a complete character or an error condition, then it emits these character transactions to both upper layer, which is the packet collection layer and also to the driver to help manage the exchange layer tasks(link initialization, error recovery and flow control). Once the packet collector gets the characters from the character encoder, then it creates SpaceWire packets out of them and emits them to the user. As a result, user has access to all the transactions from signal level, through character level and up to packet level. Therefore user can take transactions from any of the above mentioned layer and do reporting, analysis, coverage collection, debugging etc. The implementation provides user also with the predefined protocol level coverage data by using SystemVerilog covergroup structures, which can be analyzed by the user to see if all the corner cases of a SpaceWire communication was hit.

SpaceWire

SpaceWire is a data-handling network and uncomplicated to design or implement. It also provides high-speed (2 Mbits/s to 200 Mbits/s) full-duplex data links and bi-directional [1]. There are 6 layers in SpaceWire as Physical Layer, Signal Layer, Character Layer, Exchange Layer, Packet Layer, and Network Layer [1].

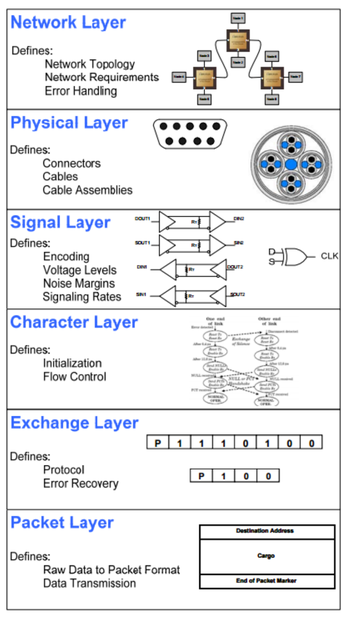

Figure 1: SpaceWire OSI Model [3]

In Physical and Signal Layers, Low Voltage Differential Signaling (LVDS) is used to execute communications such as packet and token passing accross a SpaceWire network. Also, Data-Strobe (DS) Encoded LVDS is used to communicate full-duplex, serial, and bi-directional data. In other words, DS encoding is used in SpaceWire in order to send information over LVDS. In Fig 2, data and strobe signals are XORed and synchronous clock is generated. The data values are transmitted directly and the strobe signal changes state whenever the data remains constant from one data bit interval to the next [2].

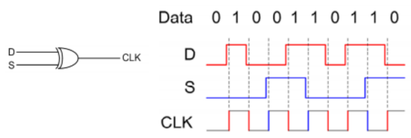

Figure 2: Data Strobe [3]

In the Character Layer, characters are defined as either link or normal characters. Normal characters pass through a SpaceWire network at a packet layer whereas link characters do not pass. In the Exchange Layer, state diagram of the initialization and error recovery sequence, shown in Fig 4, is vital for this work.

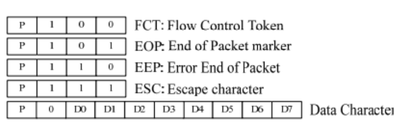

Figure 3: Character Layer [3]

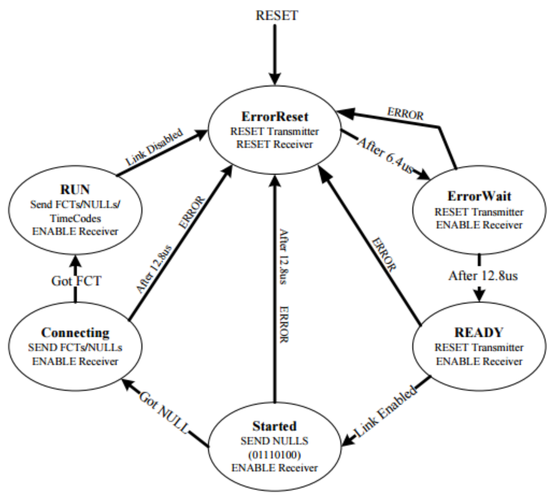

Figure 4: Exchange Layer State Diagram [3]

Universal Verification Methodology

UVM provides flexible and well established solution for complex system design verification. Since UVM consists of reusable components and is supported by tools of all major vendors of the industry, it is flexible [4]. In Fig 5, UVM testbench is shown .

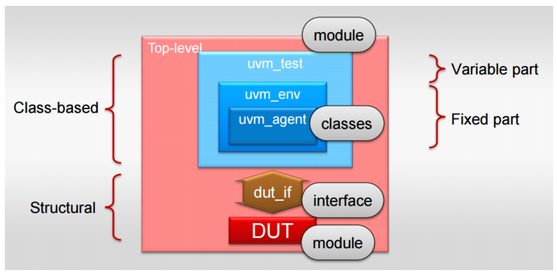

Figure 5: UVM Testbench [5]

Moreover, there are 4 phases in UVM as Build Phase, Connect Phase, Run Phase, and Report Phase. In Build Phase, child components, ports, and components are constructed and configured. In Connect Phase, ports and exports of components are connected. In Run Phase, main body of test is executed. In Report Phase, the pass and fail status are reported.

Coverage is a metric that we use it in order to measure verification progress and completeness. Coverage metrics tell us what portion of the design has been activated during simulation (that is the controllability quality of a testbench).

Implementation and Simulation Results

The implementation has been simulated with Mentor Graphics QuestaSim 10.3 version.

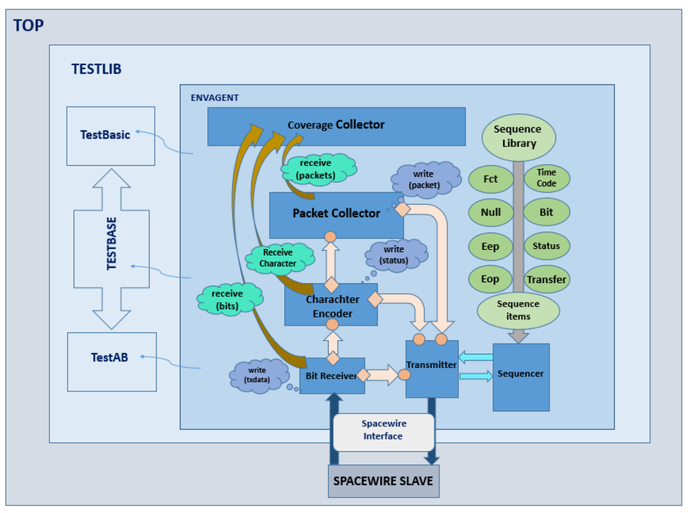

Figure 6: Overall Structure

Operational Results

Fig. 7 shows the signal level activity. As soon as Data and Strobe signals are sent within this period, they are synchronised and clock is generated. As a result, state transition is completed from beginning to the last state. Link initialization transaction level activity can be seen in Fig. 8. STARTED state is returned to the RESET state a couple of times because the length of time since the last transition on Data or Strobe lines are sent longer than timeout period (850 ns).

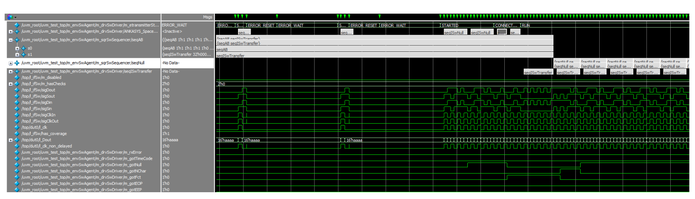

Figure 7: Clock Synchronization

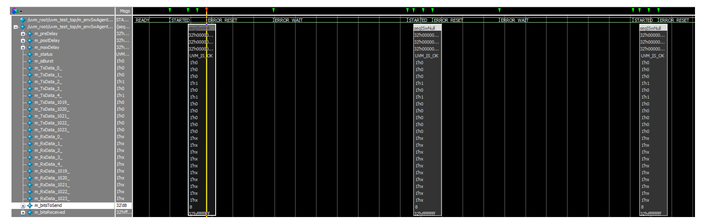

Figure 8: Link Initialization

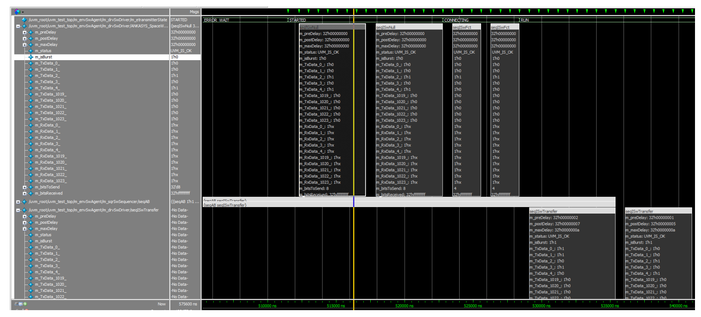

Figure 9: Started and Connect States

Fig. 9 shows the NULL and FCT character transactions in STARTED and CONNECTING state. In Fig. 10, RUN state is showed. After N_Char is sent, it stays at the RUN state by keeping on sending NULLs.

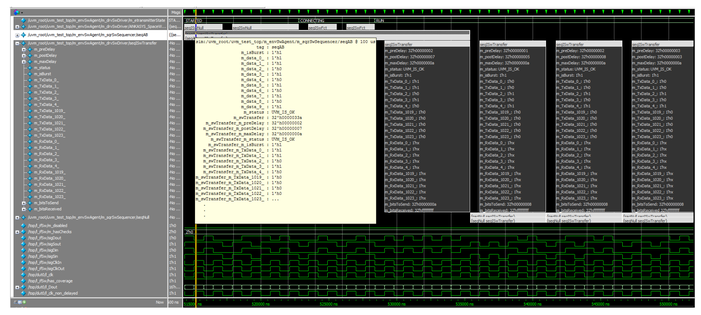

Figure 10: Run State

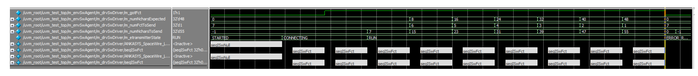

Figure 11: Credit Error

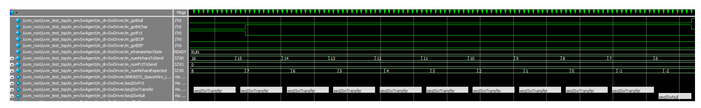

Figure 12: Credit Error

In Fig. 11, there are 9 FCTs which are sent to DUT but it causes credit error because the credit counter(N_Chars expected) shall hold a maximum credit count of 56 due to the receive buffer capacity. Also, in Fig. 12, 9 N_Chars are transmitted after a FCT has sent which results in credit error, because for each FCT receive buffer can accommodate maximum 8 N_Chars.

Fig. 13 shows coverage results. The declared covergroups can be seen explicitly in this graphic. For example, character and code covergroups have %100 coverage but Rx and credit errors have no coverage because these situation never occurs in the performed test. Multiple tests should be defined so as to take more valid results.

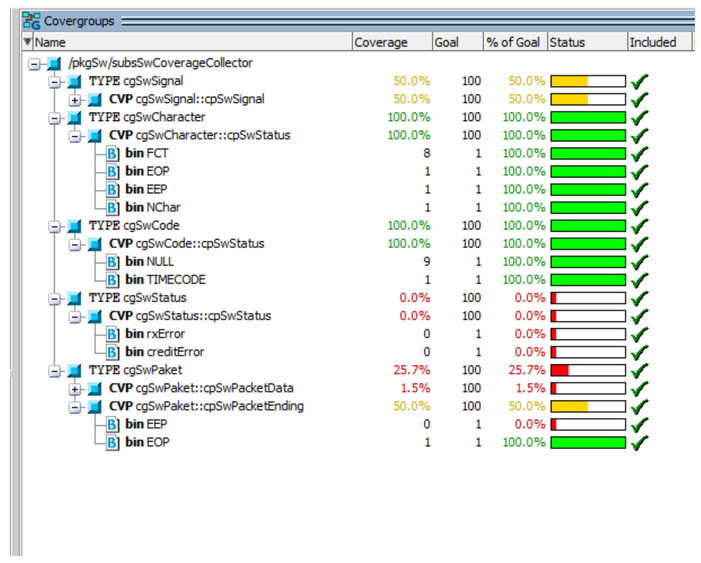

Figure 13: Coverage

References

[1] S.M. Parkes, "SpaceWire: The Standard", Proceedings, DASIA 99, Data Systems In Aerospace, 17-21 May 1999, Lisbon, Portugal, pp111-116, European Space Agency (ESA) publication no. SP-447, ISBN 92-9092-788-7.

[2] SpaceWire User's Guide (2012). [Online]. Available: https://www.star-dundee.com/knowledge-base/spacewire-users-guide

[3] Aeroflex Colorado Springs Application Note. *AN-SPW-004-002* [Online]. Available: http://ams.aeroflex.com/pagesproduct/appnotes /SpWin6PagesApNote.pdf

[4] Madan, R.; Kumar, N.; Deb, S., "Pragmatic approaches to implement self-checking mechanism in UVM based TestBench," *in Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in* , vol., no., pp.632-636, 19-20 March 2015

[5] Fitzpatrick, T. [Online]. Available: https://verificationacademy.com/cookbook