2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Protocol verification of an IEEE 802.3bw PHY

Application of mixed signal extensions to an UVM verification environment

Joen Westendorp, NXP Semiconductors, Nijmegen, the Netherlands (joen.westendorp@nxp.com)
Marcel Oosterhuis, NXP Semiconductors, Nijmegen, the Netherlands (marcel.oosterhuis@nxp.com)

***Abstract*— IEEE 802.3bw-2015 (PHY) specifies the physical layer of automotive ethernet which drives a single balanced twisted-pair medium (100BASE-T1). The PHY implementation requires analog/mixed signal circuits, pulling verification into the mixed signal domain. PHY verification cannot be completed with a common UVM test environment due to the analog circuits involved. Still a single verification environment is desired, this yields less time spent in test development and improved communication between engineers, resulting in faster time to market with better verification quality and less development cost. The problem of combining digital and analog circuits in a single UVM verification environment is addressed by introducing dedicated UVM mixed signal extensions. The application of the verification environment is demonstrated by verifying the PHY, which is part of an Automotive Ethernet system.**

Keywords—mixed signal system level, protocol verification, UVM, mixed signal extensions

I. INTRODUCTION

IEEE 802.3bw-2015, also called automotive ethernet, is the specification of the physical layer (PHY) used in ethernet based networks for automotive applications. This PHY is divided into two sublayers: the physical coding sublayer (PCS) and the physical medium attachment (PMA) sublayer. The PCS communicates with higher levels of the protocol through the media independent interface (MII). The PMA communicates with other PHY's through the media dependent interface (MDI). Also, a management data input/output interface (MDIO) is specified, managing both the PCS and PMA sublayers. [1]

The MII and MDIO interfaces are digital. An UVM test environment which uses UVM register models, is applied to exhaustively verify these interfaces. The MDI interface is analog in nature. It should also support bidirectional communication of PAM3 symbols. Both PHY's can transmit symbols at the same time, so there will be superposition. This does not allow a digital representation of the MDI, while keeping the design representations pin compatible. Mixed signal extensions are needed to include the MDI into the UVM verification environment.

Another classical approach to verify this type of circuits is to create a test bench with two back-to-back connected DUT's. The problem with this approach is that the analog functionality is black boxed. Debugging is cumbersome and time consuming when unexpected behavior is observed at one of the interfaces. It is desired that the verification environment can source and probe analog signals to create visibility, to enable error injection and to perform coverage collection in the analog domain.

UVM complies with these requirements by offering coverage collection and randomization. UVM lacks support of processing analog signals. Mixed signal extensions are proposed to add these capabilities to the verification environment.

Some results of the PHY verification will be presented based on a released Ethernet PHY [2], demonstrating the feasibility of the approach.

II. REQUIREMENTS

First, tests and test items for functional verification are defined the verification team [3]. The test items are implemented in these tests by means of assertions or checks in a scoreboard. The use cases are defined by taking the complete IC and its environment into account, therefore simulation and verification is a mixed signal challenge.

Register verification is applied for interrupt status and communication status registers of the MDIO interface. The interrupt status register has bit fields containing the status of the power state, wake up detector and the link status. The communication status register contains information about the link quality, the number of receive and transmit errors and the general state of the PHY. The complete PHY, including a 100BASE-T1 compliant medium, is required to trigger events at the physical medium. These events will update the values of the registers, demonstrating that the required functionality is correctly implemented.

Existing verification IP (VIP) is available for the verification of the MDIO interface. However, no VIP exists to verify the MDI. A UVC capable of dealing with analog signals is needed. Components and interfaces with the ability to source and probe analog signals are not provided by the UVM library.

The mixed signal UVC will control the signals at the physical medium. The requirements for such mixed signal UVC are, amongst others:

- The mixed signal UVC shall implement all protocol layers, the UVC shall contain a reference model of the analog centric PMA
- The mixed signal UVC shall generate valid traffic, according to the protocol specification, including the electrical characteristics.
- The mixed signal UVC shall provide functional coverage at all protocol layers, including the analog centric PMA
- The mixed signal UVC shall adapt to the physical medium

III. RELATED WORK

To reach the goal of performing protocol verification, two extensions to make UVM more applicable for verifying a protocol are needed. The first are the mixed signal extensions, these cover analog signal creation and sampling. The second extension is the stacked UVC extension.

Related to analog signal creation and sampling, four contributions are considered:

Verilog-AMS adapters have been proposed as custom interface elements between the DUT and the test bench. These adapters are inserted between the DUT and the SystemVerilog interface which is connected to the driver and the monitor. Also, the concept of static and dynamic configuration is defined before [4].

Another approach is to connect a standard UVC to an interface and implement API calls inside this interface to drive analog sources by means of out of module references (OOMR) [5].

Analog UVC's have also been proposed. An analog UVC drives and monitors analog signals without adapters. The analog agent generates the time domain waveform. The idea of calling an external tool for signal processing through the SV-Direct Programming Interface (DPI) was introduced as well. This is needed because the signal processing capabilities of SystemVerilog are limited. A reference model using the external code is implemented in the scoreboard, effectively comparing waveforms [6].

Another approach is to abstract the analog signals. Describing analog signals by means of their properties is describing them at the transaction level. An API to an external library is reported. Drivers and monitors are registered and configured through this API. The UVC creates and samples time domain analog signals. The UVC is configurable and extendable in this proposal [7].

The disadvantage of the first two proposals is the need for additional wrappers. These approaches are not considered to be maintainable, because updates in the adapter or interface need to be reflected in every verification environment where these instances are used.

The disadvantage of the latter two proposals is that they do not work with electrical signals. Without this constraint the concept of SystemVerilog virtual interfaces can be used, similar to a digital UVM verification environment.

Related to layered UVC's the following contribution is considered:

It was proposed to create an UVM component that make the translations between different protocol layers, instead of using layered sequences for this purpose [8]. Implementing the functionality to process data when going from one protocol layer to another in a component is more transparent than implementing this functionality in a sequence.

The proposal in this paper differs from earlier work in the sense that the mixed signal UVC works with electrical signals as well as analog signals represented by real numbers. The UVC proposed in this paper can also be connected to other UVC's, due to its strict layering. The other UVC verifies the higher protocol layers, enabling functional verification of the protocol stack. Finally, the proposed UVC does not require AMS adapters connecting to a SystemVerilog interface, reducing the complexity of the solution and improving maintainability of the mixed signal UVC.

IV. MIXED SIGNAL VERIFICATION METHODOLOGY

The protocol specification is divided in layers. The abstraction of the operations on the data increases when moving up in the protocol stack. This is done to separate concerns when creating a communication system. The verification environment will implement the layering as well, the ultimate goal is to enable system level verification. The following levels of signal abstractions are defined for the verification environment:

- Transaction level, these are ethernet frames
- Signal level, for example the individual PAM3 symbols
- Physical level, this includes voltages and currents

Applied to analog signals, this means: The signal is described by its properties in the transaction level layer. UVM catches these in sequence items. In the signal layer, the analog signal is described in the time domain by time-value pairs. This compares to logic signals in the digital domain. UVM connects to these signals through drivers and monitors. The analog signal is described by its physical quantities in the physical layer, these are the electrical signals. UVM does not have the capability to work with this type of signals.

When mapping the IEEE 802.3bw specification to these layers, it is observed that the PMA fits in the physical layer. The PCS performs operations which belong to the signal layer, while the MDIO interface can be best described at the transaction level.

Because UVM does not work with analog signals, extensions are needed to enable verification in the physical layer. The concepts which make UVM scalable and extendable need to be maintained. More specific, these are the

factory and the configuration mechanism. The use of a factory enables overriding components. Configuration allows the component to be usable in a wide number of applications.

To avoid confusion with pure digital verification approaches, new components are defined. These are the source and the probe. There is a functional equivalence between a source and a driver. They both generate signals. Also, similarities in functionality can be found between a probe and a monitor. They both sample and collect signals. Sources and probes differ from their digital counterpart in the sense that they can generate and sample electrical signals in the physical layer. For reasons of clarity, the source will override a driver in a mixed signal UVC. This will also happen with the probe and the monitor. Drivers and sources or monitors and probes will no co-exist in a single agent.

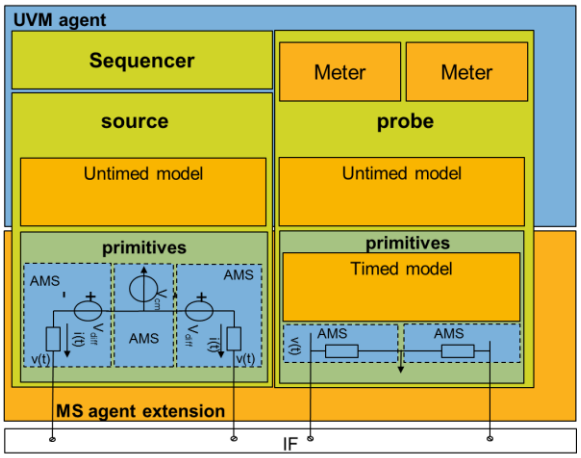
The source translates sequence items from the sequencer into analog signals. The source performs operations in all defined abstraction layers. A source receives a sequence item from the sequencer, which is a transaction level object and creates an analog signal, which is in the physical layer.

A probe samples, collects and analyses the analog waveform. The result of the analysis is a signal property. The translation from physical level to signal level and finally transaction level signals can be easily seen in these operations. Analyzing the collected samples is the responsibility of a meter. A meter is instantiated inside the probe and implements an analysis function [7].

Some analysis functions require information about the sampling properties of the analog signal. An example is the FFT: To extract a frequency, information about the buffer size and sampling rate is required. Automatically inserted interface elements cannot be used to sample a physical analog signal. Information about the A/D conversion done by the interface elements is not known to the verification environment. This information is required for certain analysis types.

There might be a need to process an analog signal before it is generated or analyzed. Both the source and probe components provide a hook to implement a model for such processing. This is somewhat equal to what has been presented before [8].

The mixed signal extensions need to connect to another VIP to implement the higher layers of the protocol. A transactor component is used to make a stacked UVC.



[A picture of the MS-UVC]

V. IMPLEMENTATION OF THE MIXED SIGNAL VERIFICATION ENVIRONMENT

The implementation of the physical layer in the source and probe in SystemVerilog poses some challenges. The whole problem of verifying analog signals started with SystemVerilog not being able to deal with this type of signal. The natural alternative modelling language when analog signals are involved is Verilog-AMS. Verilog-AMS is a Verilog derivative like SystemVerilog, enabling a mixed language setup. However, Verilog-AMS does offer the possibility to follow object oriented paradigms; Verilog-AMS instances only exist as modules. To facilitate further discussion, dynamic and static instances are defined. Dynamic instances are objects of classes, in this context implemented in SystemVerilog. Static instances are Verilog-AMS modules and SystemVerilog modules and interfaces.

The current state of verification technology requires implementation of the physical layer in static instances. This applies to mixed signal primitives like signal generators, samplers and the timed part of the model, the latter will be discussed later. These primitives are grouped and wrapped in a parametrized SystemVerilog class. The definition of this class needs to be in the same compilation scope since it interacts with the tasks and variables of the primitives mentioned before. The class type will be stored in the configuration database and is fetched by the agent during the build phase of the environment. This configuration item is used to do a factory override of the original driver and monitor instances in the agent. As such the wrapper classes are registered in the UVM hierarchy, as the source and probe components. This mechanism is similar to setting and getting a virtual interface, which is common practice in a pure digital UVM based verification environment.

Dividing the implementation of the mixed signal verification environment into static and dynamic instances calls for a redefinition of the configuration mechanism in the verification environment. As stated before, configuration makes a verification environment reusable, extendable and scalable. The concept of static and dynamic configuration [4] is used. However, dynamic configurations are always preferred over static configurations. Pushing the class implementation close to the AMS primitives brings the UVM configuration mechanism to the physical layer, extending the reach of dynamic configuration.

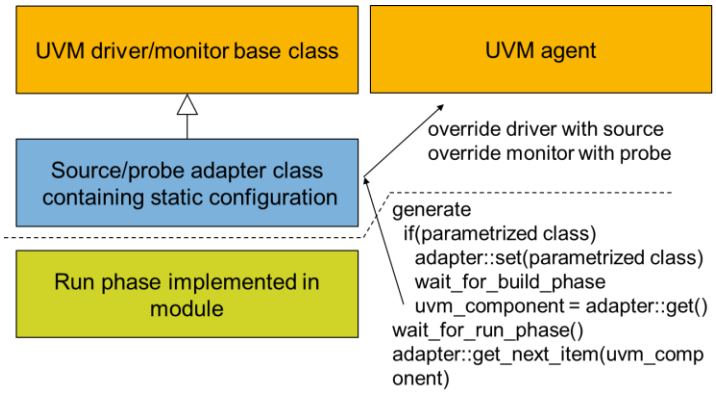
SystemVerilog classes cannot instantiate static instances, therefore the instantiation of static instances cannot be a dynamic configuration item. The instantiation of static instances must be implemented as static configuration item. Multiple mechanisms can be used to implement static configurations. Examples are Verilog configuration, parameters or the use of conditional generate blocks. This proposal uses a combination of Verilog configurations and the conditional generate block. The Verilog configuration is used to select AMS instances with the desired discipline. Multiple implementations of an AMS instance with different port types exist, examples are Verilog-AMS instances with electrical ports, SystemVerilog instances with real nettype ports or with user defined nettypes.

The conditional branches in the generate block can be selected using parameters when instantiating the mixed signal extensions in the netlist. To prevent an unlimited growth of such parameters, they will be declared as localparam in a SystemVerilog class. This makes these parameters available by referencing to the class, such that they can be used as a condition in the generate block.

SystemVerilog classes cannot be declared inside a generate block, this means that the class definitions of the sources and probes need to happen in a separate package. It is intended to move all UVM functionality out of the generate block into the adapter package. The adapter is not in the same compilation scope of the AMS instances which are instantiated inside the conditional branches in the generate block. Consequently, the implementation of the run phases in the source and probe cannot be at the same place as the class declaration. The class declaration itself is still needed for overriding the driver or monitor in the mixed signal UVC.

Fortunately, the uvm_phase base class provides a wait_for_state method. This method can be used to implement the run phase inside a module. After a reference to the dynamic source instance is obtained, a forever block is implemented containing the get_next_item call to fetch the sequence item and specific code to control the AMS instances to generate and sample analog signals.

This implements the equivalent of a factory for static instances. This configuration mechanism makes the static configuration scalable, extendable and maintainable.



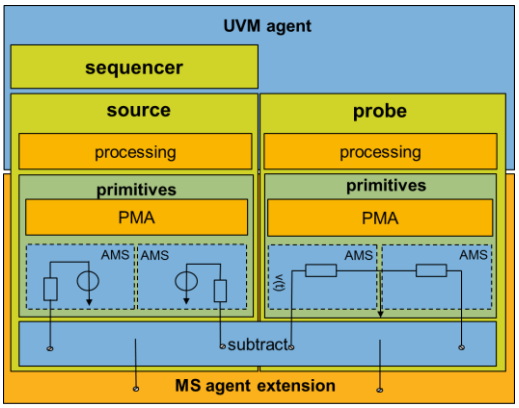
[a picture to show static configuration]

The model to perform data processing on analog signals has been split into a timed model and an untimed model. The timed model is implemented in a static instance, it's instantiation is controlled by the static factory. The timed model can process data taking time into account, it can process individual samples. Examples are frequency mixers, etc. The untimed model is implemented in a dynamic instance and can be implemented by implementing a callback in the source or probe. A vector of data is processed by the untimed model at once, time is not taken into account. An example of usage of the timed and untimed model is given in the paragraph below.

The above concepts shall be used in the PHY verification, the source and probe configurations will be explained in more detail: An untimed model translates the symbols vectors received from higher level protocol layers into symbols. A timed real number model synchronizes the symbols with the clock. DC voltage sources will generate the analog PAM3 signals, based on the output of the timed model. The PAM3 signals will then be applied to the physical medium.

At the probing side, a sampler connected to the physical medium samples the analog signal. The samples are sent to a real number model implementing the PMA receive function, including clock recovery. The PMA receive functions translates the individual samples into symbols. The symbols at the output of the timed model are passed to the probe to be collected. The collected symbols are then sent to an untimed model. This untimed model extracts symbol vectors from the collected symbols. Also, analysis and coverage collection is performed on the received symbols. The retrieved symbol vectors will be further processed in higher protocol layers.

Both PHY's can drive the physical medium at the same time. This results in superposition of the symbols on the physical medium. The PHY needs to subtract the symbol it sends from the voltage seen at the physical medium to extract the received symbol. Another timed model will be inserted between the source/probe and the physical medium to implement this functionality.



[Picture of the MS-UVC and the mapping of above concepts to the DUT]

VI. APPLICATION

The described is used to verify the pre-defined use cases, verification of use cases is divided into tests. One of the tests derived from the use cases requires the following sequence:

- Power the DUT
- Configure the DUT as master
- Establish a link with the slave
- Set the signal amplitude
- The UVC sends/receive random data with randomly injected errors
- Read from the DUT
- Reset the DUT

Repeat the above for all signal amplitude settings

The following test items are implemented in the verification environment:

- The data received at the MII interface shall be equal to the data sent by the connecting device
- The data sent through the MII interface shall be equal to the date received by the connecting device
- An error shall be flagged if data cannot be received

A scoreboard is used to implement the listed test items. The scoreboard is not part of the UVC, but a separate component. The scoreboard resides in the transaction level layer. As a consequence, it only compares properties of analog signals. The analog signals need to be passed through a meter before they can be used be the scoreboard, because meters perform analysis to extract properties of analog signals.

Subscribers are used to implement the functional coverage collectors. They are connected to the output of the untimed model and track coverage information about the symbols on the physical medium. Higher level VC's collect functional coverage according to the protocol layer they are intended to verify.

[show the virtual sequence executing the above]

[show coverage metrics of analog signals]

VII. CONCLUSION

A mixed signal UVM verification environment for protocol stack verification has been presented. A layered UVC is proposed, enabling mixed signal verification. An extendable static configuration mechanism is proposed, bringing UVM concepts to analog/mixed signal verification. It is demonstrated on a real product that the resulting mixed signal UVC can be used to verify the complete protocol stack.

[1] IEEE Std 802.3bw-2015, https://standards.ieee.org/findstds/standard/802.3bw-2015.html, 2015

[2] TJA1100 datasheet, https://www.nxp.com/docs/en/data-sheet/TJA1100.pdf

[3] W. Tibboel, H. Shuang, H. van Orsouw, "An efficient requirements-driven and scenariodriven verification flow", DVCon Europe 2017

[4] I. Ignjic, Connecting UVM with mixed signal design, DVCON 2017.

[5] A. Freitas, R. Santonja, UVM Ready: Transitioning Mixed-Signal Verification Environments to Universal Verification Methodology, DVCon Europe 2014

[6] J. McGrath, e.a., Comprehensive AMS Verification Using Octave, Real Number Modelling and UVM, DVCon Europe 2015

[7] A. W. Rath e.a., Towards a UVM-based Solution for Mixed-signal Verification, DVCon Europe 2016

D. Cornfield, The Universal Translator, A fundamental UVM Component for Networking Protocols, DVCon Europe 2014