AMOTIVE

Proper Probing:
Flexibility on the TLM Level

Gergő Vékony

Motivation

Complex modules sometimes call for white box test approaches, as there may be some internal variables and signals that cannot be reached from the module's outermost surface. One such test method is probing where a testbench (TB) related component is bound into the Device Under Test's (DUT) structure to provide information about its internal activities.

There are several probe methodologies employing the Universal Verification Methodology (UVM) that provide different kinds of flexibility and configuration options. As the probed values are usually propagated via Transaction-level Modelling (TLM) transactions to other UVM components, code maintainability, reusability and probe–monitor decoupling are strongly considered factors in choosing most suitable implementation.

Conclusion

This generalized interface technique has been well suited and applied to a highly complex neural network related automotive design. The ability to have an array of probes deployed with different configurations with minimal modification of the verification environment saved considerable amount of time and effort of the verification team.

Such an approach provides clear benefits against conventional parametrized solutions, especially if horizontal and vertical reuse is considered.

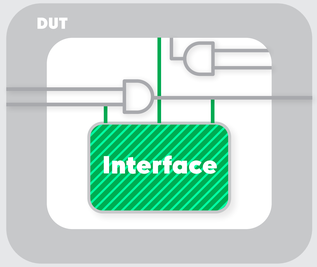
Methodologies

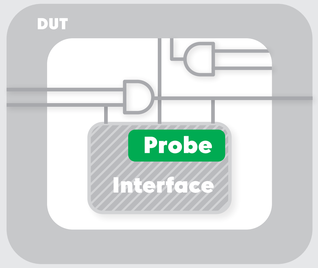
Interface binding

- An interface with an appropriate port list is bound to the DUT.
- Ports serve as access points between the testbench and the DUT.
- A compilation-unit scope hosts the binding and the ConfigDB access.
- The testbench accesses the bound interface like any other virtual interfaces.

Polymorphic class access

- A widely available (abstract) superclass is created for the probe.
- An interface with an appropriate port list and a descendant class definition and instance is bound to the DUT.
- This class instance can access every property of the interface.
- A compilation-unit scope hosts the binding, the interface itself hosts the ConfigDB access.
- The TB accesses the descendant class instance like any other ConfigDB asset.





Challenges

Probe access

Class based probes are communicating with the testbench via a getter API or variable assignments that need to be implemented. Interface based probes may expose their full port list, without restrictions.

Quality

Different probes shall use a similar way of communication, coverage collection and report generation to ensure comparable results across the design.

Reusability

Probe functions shall be implemented with a level of abstraction in mind to ensure that the probe is – at least horizontally – reusable in regression tests.

Version control

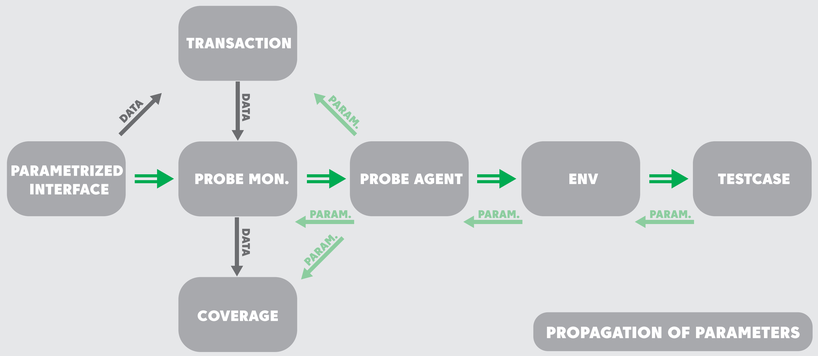
The testbench shall follow the changes in the DUT structure but also be decoupled from it. Especially in the case of probes, as binding directly depends on the DUT hierarchy.

Parametrization Drawbacks

Implementing specialized probes – and probe monitors – without parameters is a bad idea, as it may cripple reusability and result in a ***lot of code*** for maintenance.

Implementing probes with parameters may help extension and reuse, but also requires a great amount of pre-planning, as parametrization need to propagate through the (whole) TB infrastructure, resulting in ***complex code.***

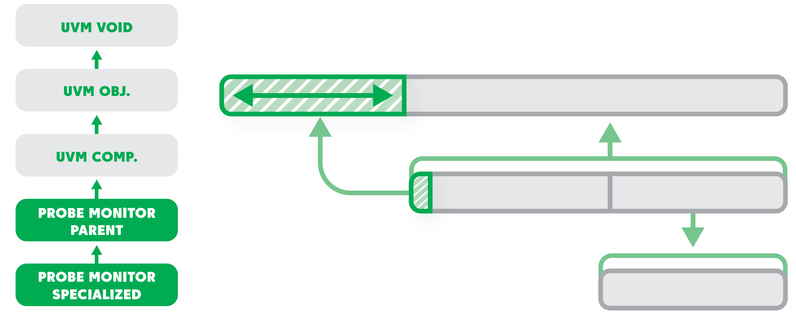
Clearly, neither of these solutions is suitable in a ***simple*** and ***widely reusable*** implementation.



A Suggested Solution: Generalization

Generalized probes are based on the *„one size fits all"* or *„maximum footprint"* approach.

- Probe infrastructure complexity and extent are traded for runtime memory footprint increase.
- One set of parameters may be present to optimize the probe infrastructure, but can be omitted entirely.
- Up to its maximum size, every signal can be handled by the probe infrastructure via assignment ***truncation and extension.***
- There is little to no need to use a vast array of different probes, thus maintenance effort is reduced.
- Probe monitors can retain compatibility and interchangeability for UVM type or instance overrides if they derive from a common superclass.
- ***Simplified probe and probe-monitor components*** encourage horizontal and vertical reuse.



Generalized Interfaces or Classes

Generalized interfaces have the following advantages:

- The full probe functionality could be encapsulated in the probe monitor, thus allowing flexible, class based handling.
- As the interface ports can be classified as variables with the ***var*** keyword, they can be passed natively as references for tasks and functions.
- This allows direct signal gating and timing functionality to be implemented in the probe monitor without additional requirements.

Generalized class based probes have the following advantages:

- A stricter API may be used within the probe. (Enforced by the abstract superclass.)
- Functionality might be separated between the probe and the probe monitor for reuse purposes.

Note: signal and clock gating is possible in classes, with dedicated variables and assignment statements for example.

A Suggested Infrastructure

The suggested infrastructure consists of the following components:

- *A generalized, variable based, maximal footprint interface:* to avoid parametrization altogether and enable timing control in the monitor component, thus reducing the bound interface – and low level probe components – to the bare minimum.
- *Bind statement wrapper structures around the DUT:* to guarantee DUT and TB version independent tracking and decoupling option for the bind statements.
- *A specialized UVM probe monitor component:* that is a descendant of an UVM probe monitor superclass to ensure factory override and interchange capability.
- *(Optionally) A bind and access pattern convention:* to avoid signal clutter and mismatches during the operation.

A testbench structure like this keeps the code complexity at an acceptable level, as only one interface, a parent UVM component and several specialized UVM components are required.

Coverage collection and TLM connections can also employ the maximum footprint sizes for their variables to maintain compatibility with the probe infrastructure and reduce the parameter induced complexity.