Verification Academy Patterns Library

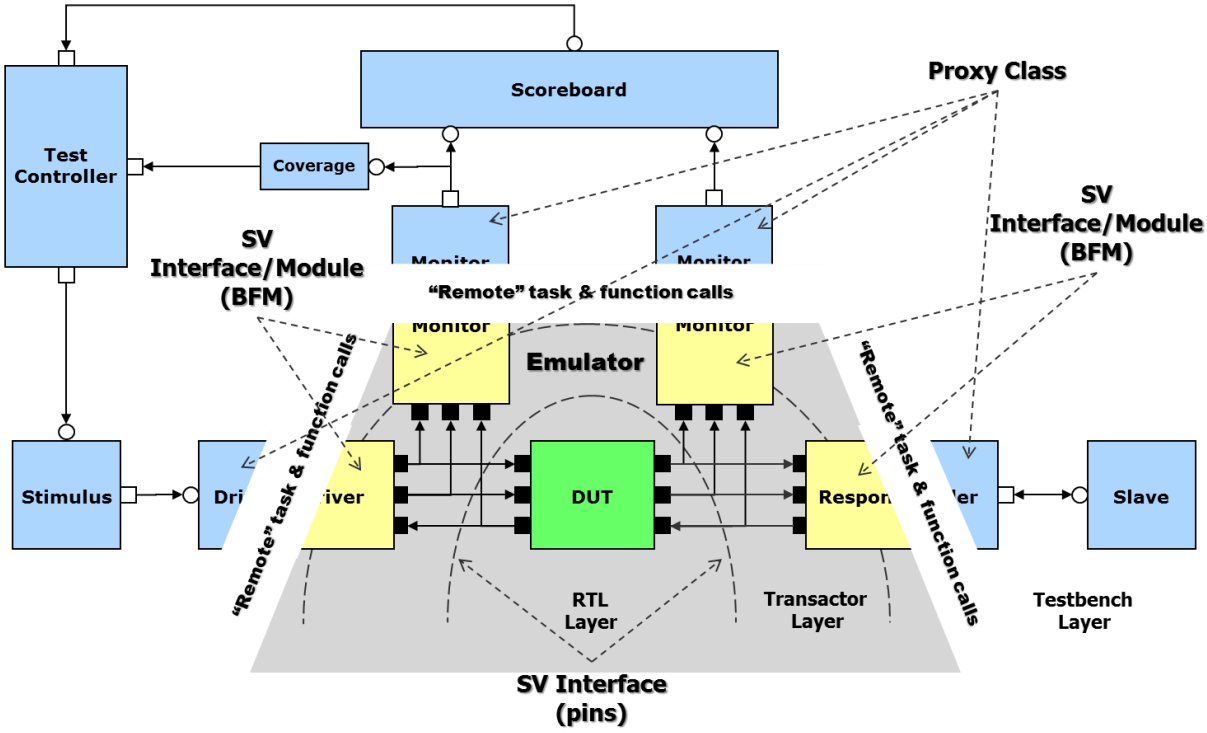
Pattern Name: *The BFM-Proxy Pair Pattern*

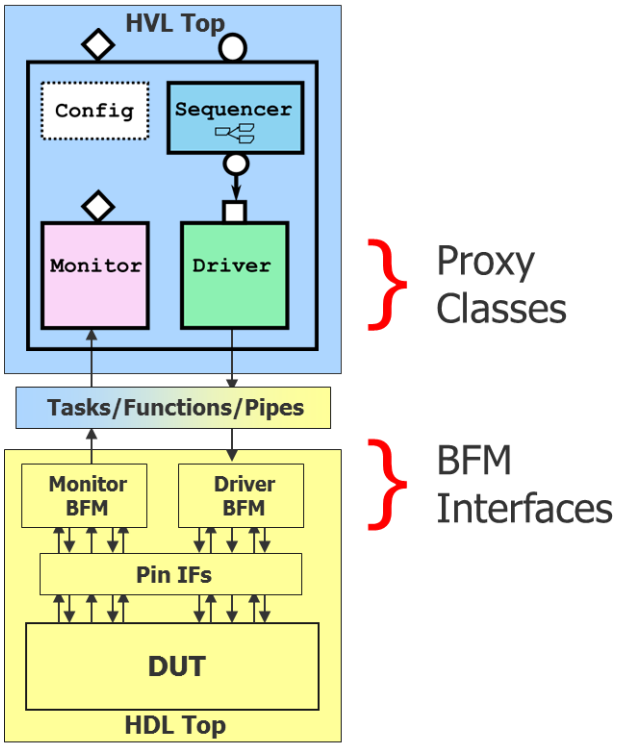
- **Intent:** The BFM-Proxy Pair Pattern is an Environment Pattern to facilitate the design of transactors like drivers and monitors for dual domain partitioned testbenches that can be used for both simulation and emulation, and across verification engines (or platforms) in general.

- **Motivation:** In order to enable and promote a verification process that is abstracted from underlying verification engines, particularly a software simulator and a hardware emulator, modern testbenches should exhibit (from conception) a dual domain architecture with partitioned HVL and HDL module hierarchies targeted for the simulator and emulator, respectively, and linked together to run in unison. Fundamental to this architecture is the employment of BFM-proxy pairs to devise so-called split transactors, where components in the HVL domain typically implemented as classes act as proxies to BFMs implemented as interfaces or modules in the (synthesizable) HDL domain. An HVL proxy provides a surrogate or placeholder for the associated cross-domain HDL BFM to control access to it via a transaction-based HVL-HDL communication model using remote function and task calls. Effectively, the proxy embodies the transactor API to upper testbench layers, abstracting the cross-domain communication and the implementation details of the BFM's bus cycle state machines.

- **Applicability:** The BFM-Proxy Pair Pattern is applicable in any situation demanding a common dual domain partitioned testbench architecture (i.e.,

separated HVL and HDL module hierarchies) for both simulation and emulation, and across verification engines in general.

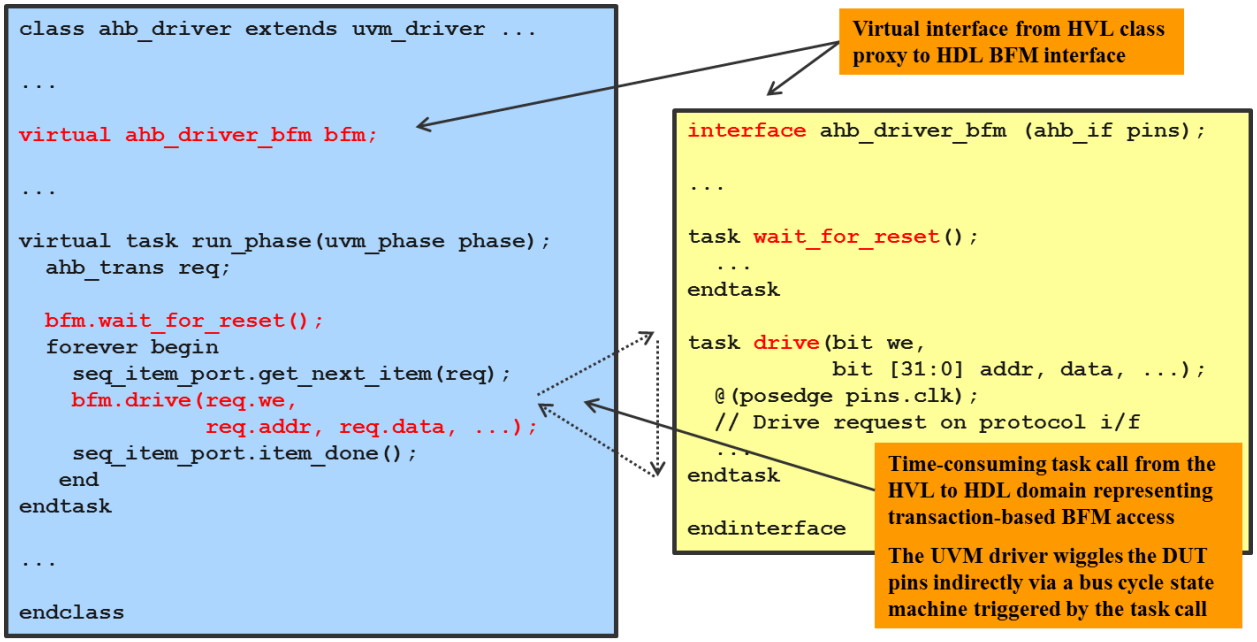


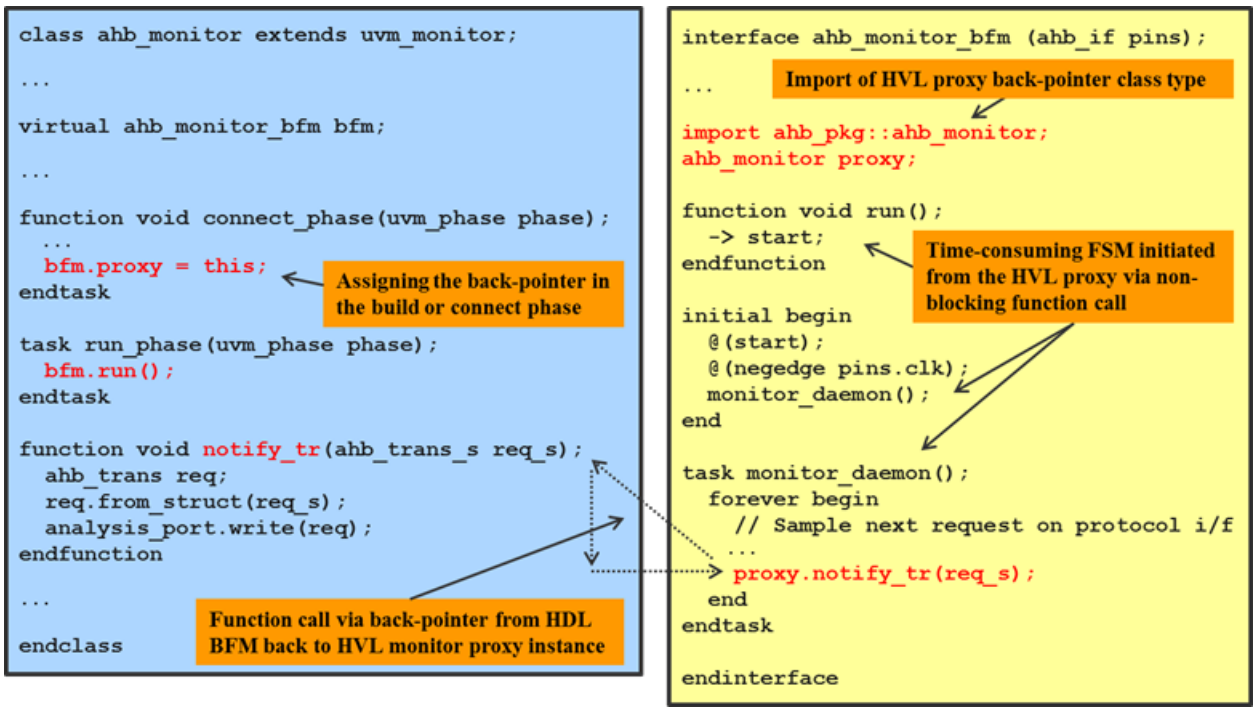


- **Structure:** The diagrams above illustrate the dual domain testbench architecture and the according UVM agent structure, respectively, with the transactors depicted as BFM-proxy pairs.

- **Implementation:** A transactor following the prescribed BFM-Proxy Pair Pattern implements a BFM as a SystemVerilog interface (or module) with dedicated functions and tasks to be called from a class proxy through a virtual (or DPI-C) interface to execute bus cycles, set parameters, or get status information. Additionally, a BFM interface (or module) may call functions defined in the class proxy via a proxy object back-pointer mechanism to provide notifications of transactions and other interesting events and conditions for control and analysis. Transaction-based cross-domain communication is thus enabled in both directions with either the HVL proxy or the HDL BFM as initiator. Each proxy-BFM pair is regarded as a joint pair representing a single transactor.

- **Example:** BFM-Proxy Pair Pattern source code examples for a UVM driver and monitor:





- **Consequences:** The dual domain partitioned testbench architecture enabled by this BFM-Proxy Pair Pattern offers maximum leverage of established simulation-based verification practices into emulation, including the benefits of using SystemVerilog and UVM for creating modular, reusable verification components and environments.

- **Related Patterns:** A precursor to this BFM-Proxy Pair Pattern is the Dual Domain Hierarchy Pattern, which advocates the HVL and HDL domain partitioning as a sound and necessary separation of concerns fundamental to emulation and other hardware-assisted verification platforms. Additionally, the BFM-Proxy Pair Pattern resembles the proxy pattern as one of the structural patterns of the Gang-of-Four's OOP design patterns (though applying instead between a dynamic proxy object and a static interface or module).

- **Contributor:** Hans van der Schoot, based on the following works:

  https://verificationacademy.com/cookbook/emulation

  https://verificationacademy.com/cookbook/emulation/splittransactors

"*UVM & Emulation: How to Get Your Ultimate Testbench Acceleration Speed-up*," H. van der Schoot and A. Yehia, in Proc. of DVCon Europe 2015, Munich, Germany, November 2015

"*Off to the Races with Your Accelerated SystemVerilog Testbench*," H. van der Schoot, A. Saha, A. Garg and K. Suresh, in Proc. of DVCon 2011, San Jose, CA, USA, March 2011

- **Release Date:** February 25, 2016

Corrections and Suggestions: *To improve the quality of the Verification Academy Patterns Library we welcome your comments, suggestions, and corrections. Please contact us at: https://verificationacademy.com/contact*