Portable Stimulus Driven SystemVerilog/UVM verification environment for the verification of a high-capacity Ethernet communication bridge

Andrei Vintila, AMIQ Consulting
Ionut Tolea, AMIQ Consulting
Introduction

• System
• Requirements
• Traditional Approach
• PSS Approach
Example System
Recap of system features

• Bidirectional traffic:
  – TX
  – RX Legal
  – RX Illegal

• Required configuration: MAC configuration, DMA Configuration

• Control block: Configured/Not configured
  – Enables the TX traffic capability of the system

• Filtering block: Configured/Not configured
  – Enables the system capability of discarding illegal traffic
Requirements

• Verify all data paths
  – Different types of packet characteristics
  – Different types of configuration for the blocks
  – Different blocks configured / Different interfaces driven

• Ensure the re-usability of the TB
Traditional approach

- Re-use VIPs between TBs
- Re-use TBs between projects
- Re-use infrastructure from block level TBs
- Create individual testcases for each interesting scenario
Issues

• Creating dependencies between TBs
  – Different layers of integration
  – Missing good management of what can be re-used
    • Compatibility between block level TBs
    • Block level specific implementation
• A huge number of testcases implemented
  – Small deviations in traffic/configuration -> New scenario
  – Directed testcases -> Non-reusable logical layer sequences
Idea behind PSS approach

• Break the functionality of the system into modular actions
• Offer a software view when simulating RTL
  – A system feature translates into multiple software actions
  – Each software action has a correspondent in hardware stimuli
  – Hardware stimuli take a different form on each layer of integration
• The PSS model puts constraints on the implementation quality
  – Increase re-usability
  – Lower debug time
PSS as an abstraction layer

- Better control for directed scenario definition
- Better planning for verification strategy
- Ease testcase portability across multiple layers of integrations and platforms
Guidelines

- Encapsulate run-time configuration in actions
- SystemVerilog/UVM sequence layering should be used to bridge gaps between transaction level sequences and system level actions
- Synchronization and timing should translate into actions on PSS layer and event triggers on SV layer
- Coverage and logical constraints -> PSS Layer
- Protocol constraints -> SV Layer
enum line_rate_e {l_10G, l_25G, l_100G};
enum cfg_interface_e {SERIAL, PARALLEL, HIGHSPEED};
typedef bit[47:0] uint48_t;

// Display controller definition
component master_c {
    action mac_config {
        rand line_rate_e line_rate;
        rand cfg_interface_e cfg_interface;
        rand bool enable_rx;
        rand bool enable_tx;
        rand bool enable_tx_fc;
        rand bool enable_rx_fc;
        rand bool strip_header;
        rand bool check_crc;
        rand bool pause_watermark_high;
        rand bool pause_watermark_low;
    }
}

component master_cpu_c {
    action config_0 {
        rand int in [2..5] arg;
        exec body SV = """\n            config_0 seq = config_0::type_id::create("seq_0");\n            seq.local_arg = {{arg}};""";
    }

    action setup {
        activity {
            sequence {
                do config_0;
                do config_1;
                parallel {
                    do config_2;
                    do config_3;
                }
            }
        }
    }
}

Generated Scenarios 1
Generated Scenarios 2

START

a_2: dma_config
    pss_top.master

a_3: mac_config
    pss_top.master

a_5: filter_config
    pss_top.master

[parallel]

a_7: send_rx
    pss_top.master

a_8: send_rx
    pss_top.master

a_9: send_rx
    pss_top.master

a_10: send_rx
    pss_top.master

a_11: send_rx
    pss_top.master

a_12: send_rx
    pss_top.master

[parallel end]

END
Project Application

- Replace directed testcases with PSS based test generation
- Ease debug with scenario flow view
- A tighter infrastructure for verification which favors re-use
Conclusions

• PSS standard has all the necessary features to accommodate a higher abstraction layer over a SV/UVM environment.
• Due to extensive support for PSS tools, this approach bridges the communication gap between verification engineers and system architects
• A general recipe can be defined for VE development flow
Questions