Architecture optimization and design verification of the Timepix3 and the Velopix pixel ASICs

14.11.2013

Tuomas Poikela (TUCS/CERN)

SV/UVM Mini workshop

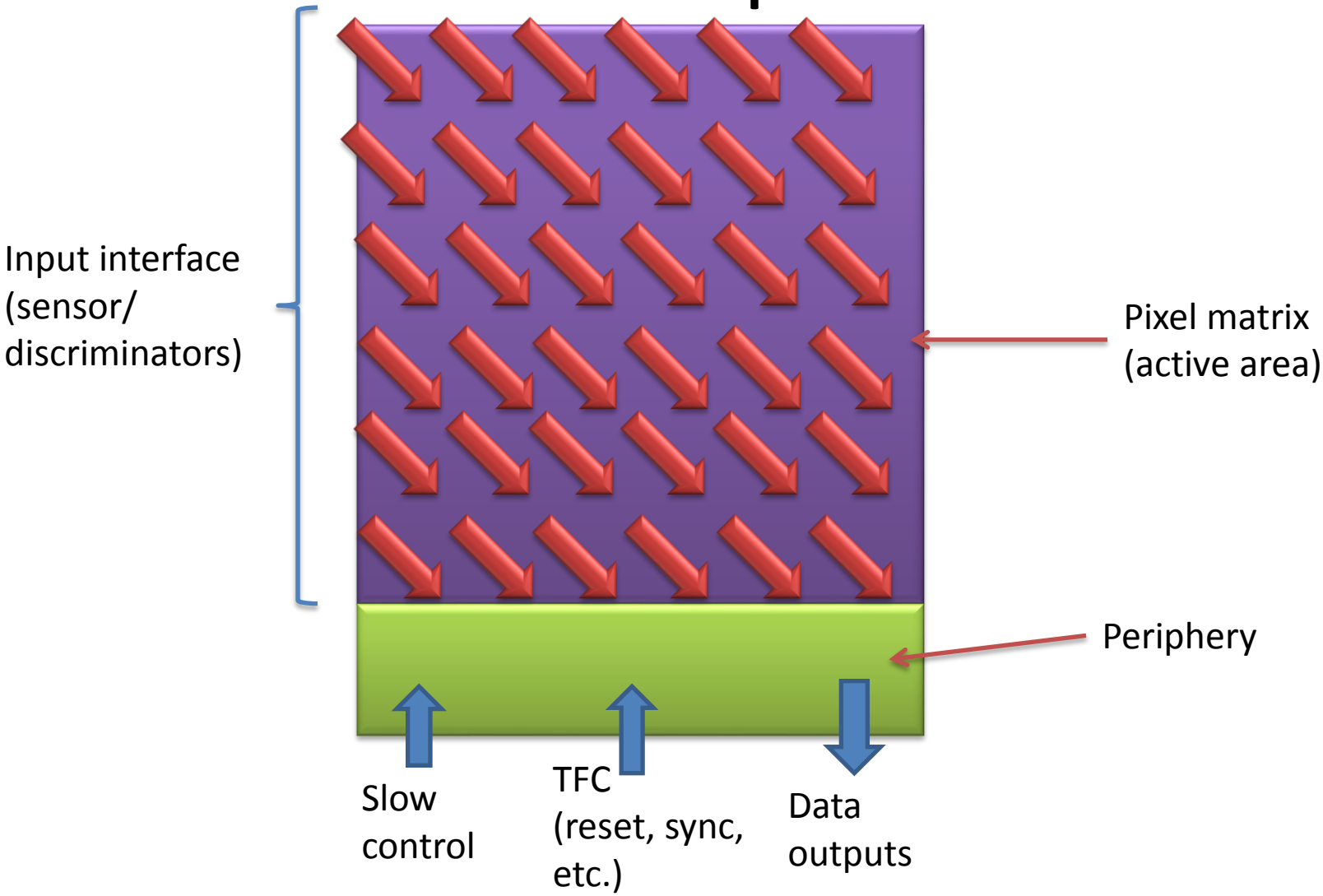
Outline

- Introduction (Pixel chip & TLM)
- Architecture optimization
  - Transaction level model (TLM)
  - Integration with verification environment
- Design verification
  - Reusing modeling framework
  - Testbench architecture
  - Assertions

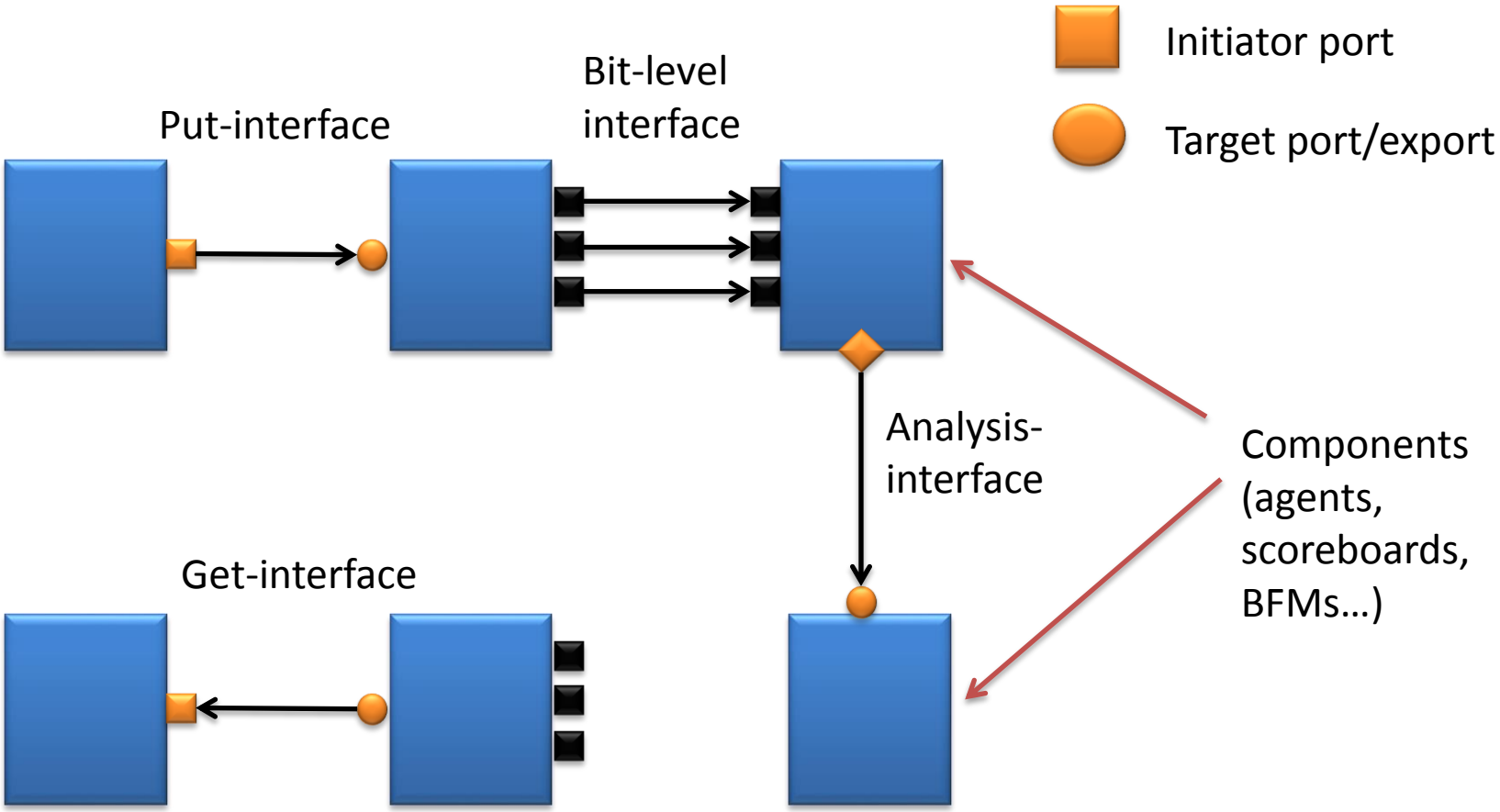
Introduction

- Techniques presented here used in two pixel ASIC projects: Timepix3 and Velopix
- 256x256 pixel ASICs with 55um x 55um pitch, designed using 130nm CMOS
- Timepix3 has been manufactured and tested to work (characterization not fully done yet)
- Velopix is still a work in progress, but the design is very well advanced

Pixel readout chip interfaces

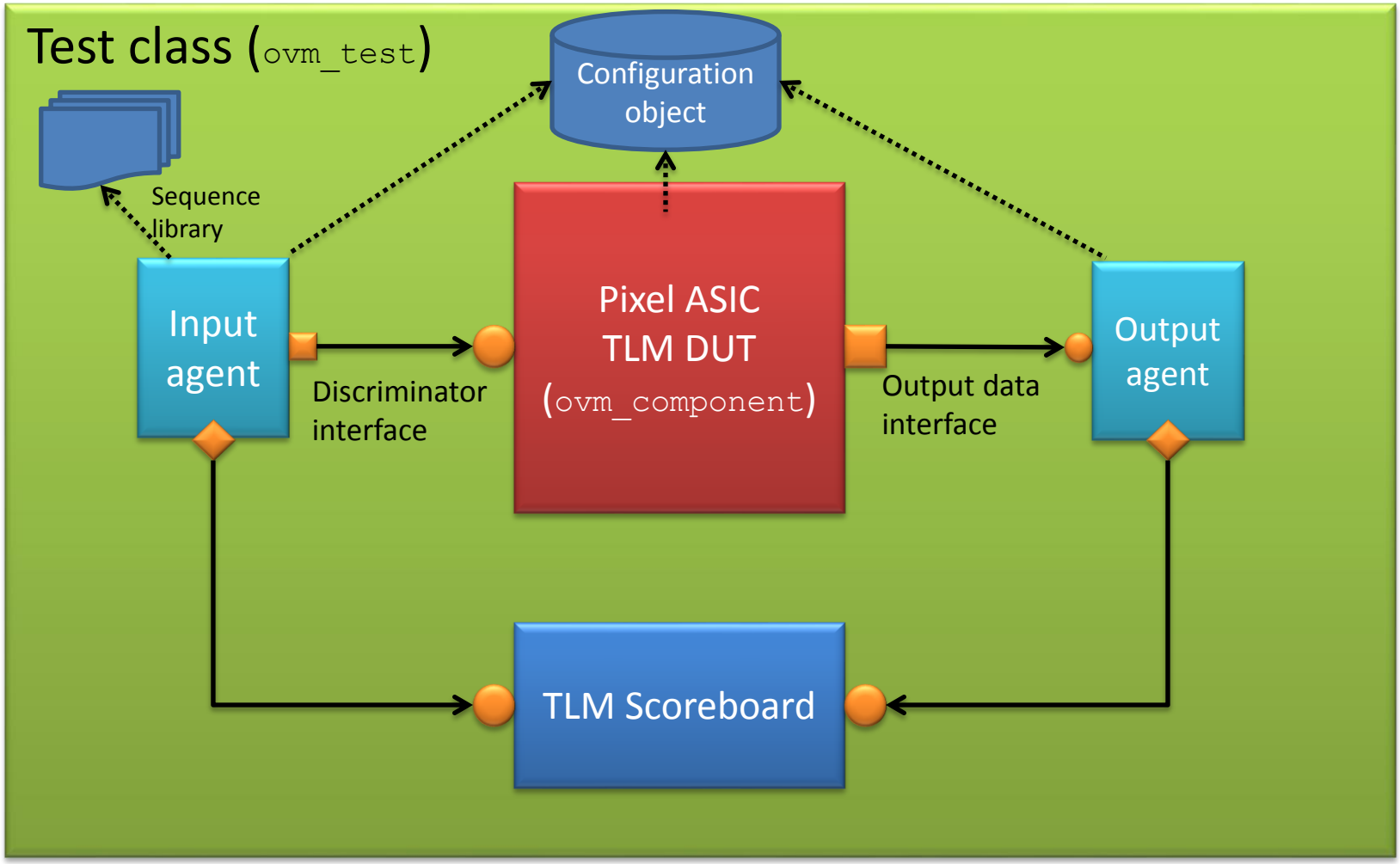


TLM notation

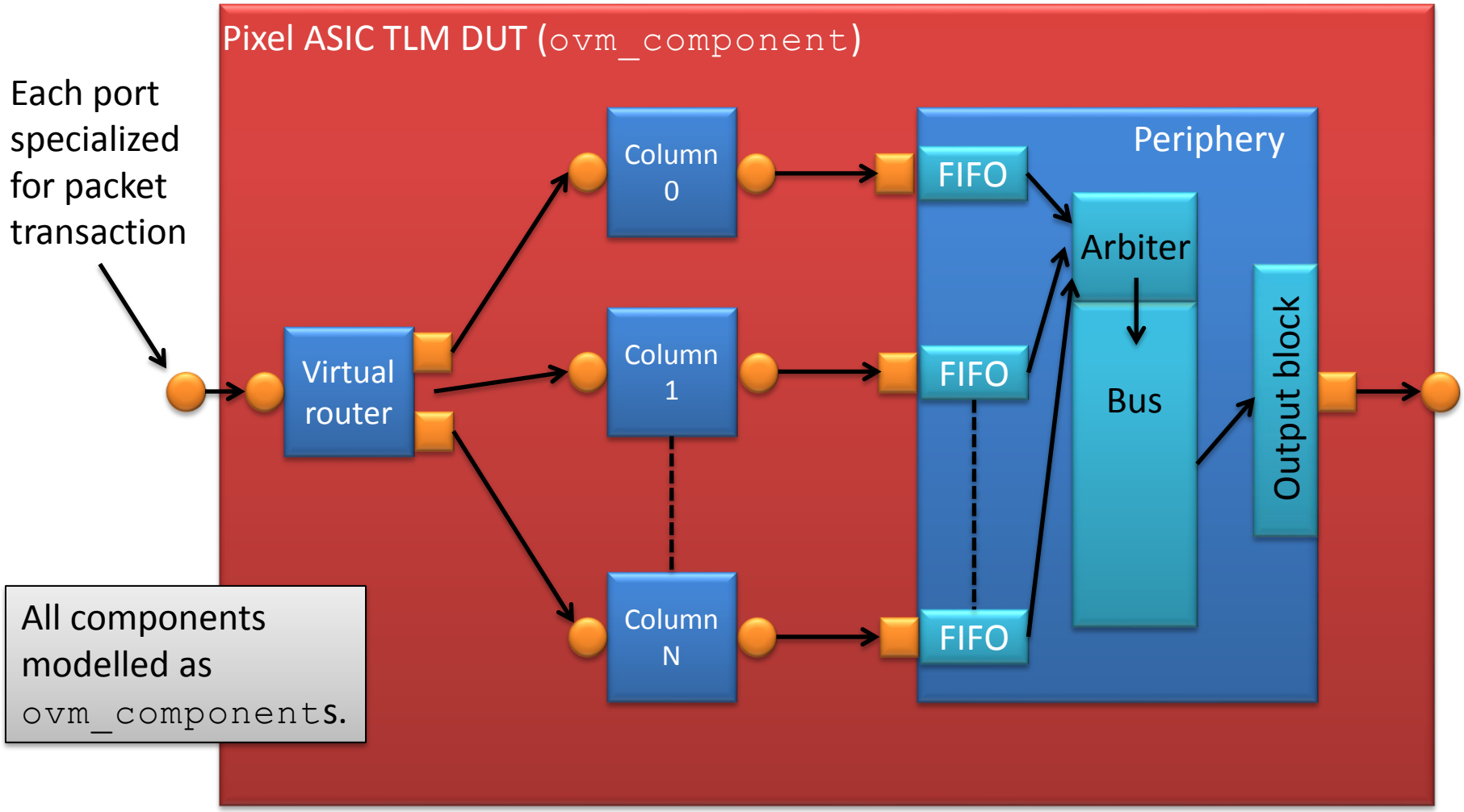


ARCHITECTURE OPTIMIZATION

Testbench for optimization



Pixel ASIC TLM



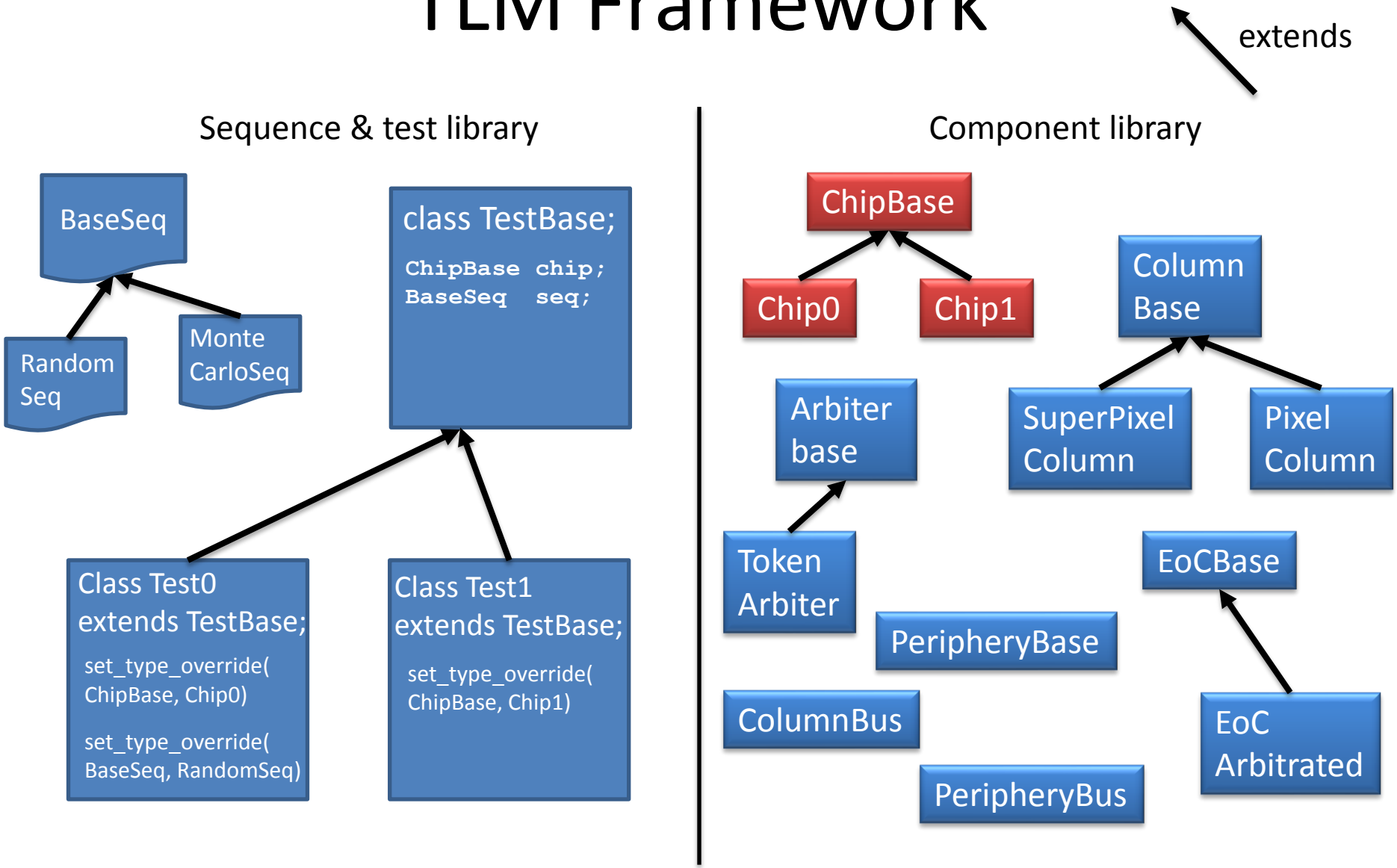
Not all TLM ports/components shown inside the periphery.

Transaction definition

```
 *
 * @class TPX2PacketTransaction
 */
class TPX2PacketTransaction extends ovm_transaction;

    //-----------------------------------------------------
    // DATA MEMBERS
    //-----------------------------------------------------
    rand  int    pixel_address_m;    ///< Address of the pixel packet.
    rand  int    size_m;             ///< Size of the packet in bits.
    rand  int    time_stamp_m;       ///< Indicates when rising edge of discriminator arrived.
    rand  int    fast_time_stamp_m;  ///< Value of the fast time stamp.
    rand  int    tot_value_m;        ///< ToT (energy) value of the hit.

    // VALUES FOR DEBUG/SIMULATIONS (cannot be extracted from RTL DUT)
    int    end_time_stamp_m;           ///< Indicates when the packet was sent off the chip.
    int    time_stamp_falling_edge_m;  ///< Indicates when the falling edge of discriminator arrived.
    int    eoc_time_stamp_m;           ///< Indicates when the transaction arrived at end of column.

    /// @cond
    `ovm_object_utils(TPX2PacketTransaction)
    /// @endcond
```

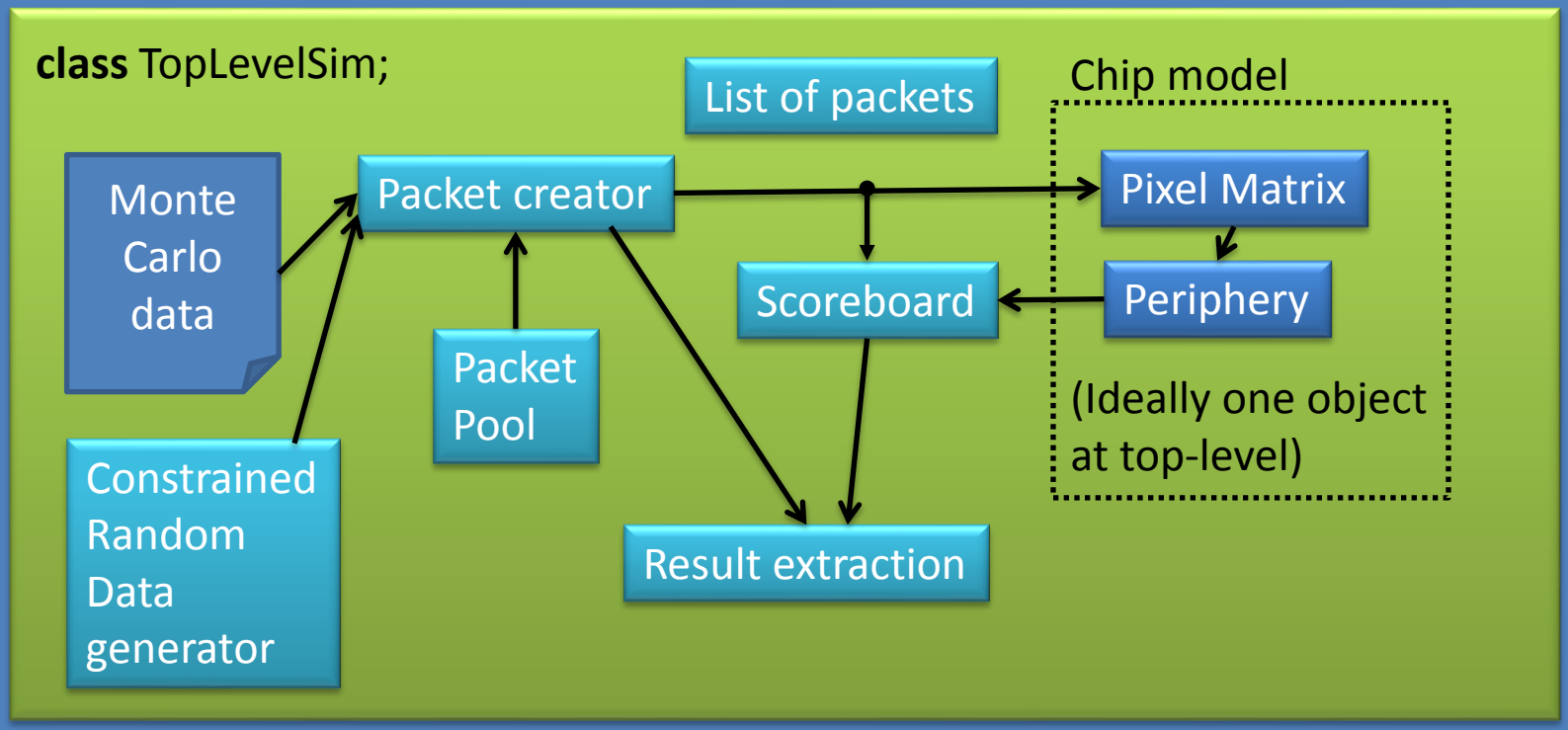
Each TLM port in TLM model specialized with **`TPX2PacketTransaction`**

TLM Framework



Different modeling approach (non-TLM/OVM)

```
module TopLevel_tb;
  TopLevelSim simulation = new;
  initial simulation.run();
```



Simulation main loop

```
VPXPacket packets[int];

virtual task run();
    for(int i = 0; i < cycles; i++) // This loop runs at 40MHz
    begin
        get_next_packets(packets);

        matrix_m.add_packets(packets);

        // Readout from matrix slower than 40 MHz
        if( (bunch_id % COLUMN_READOUT_FREQ) == 0) begin
            matrix_m.remove_packets(packets);
        end

        eoc_blocks_m.add_packets(packets);
        matrix_m.write_back_packets(packets);

        repeat(2) begin // Loop runs at 80 MHz
            simulate_output_links(packets);
        end

        finalize_packets(packets);

        ++bunch_id;
    end
    stop();
endtask: run
```

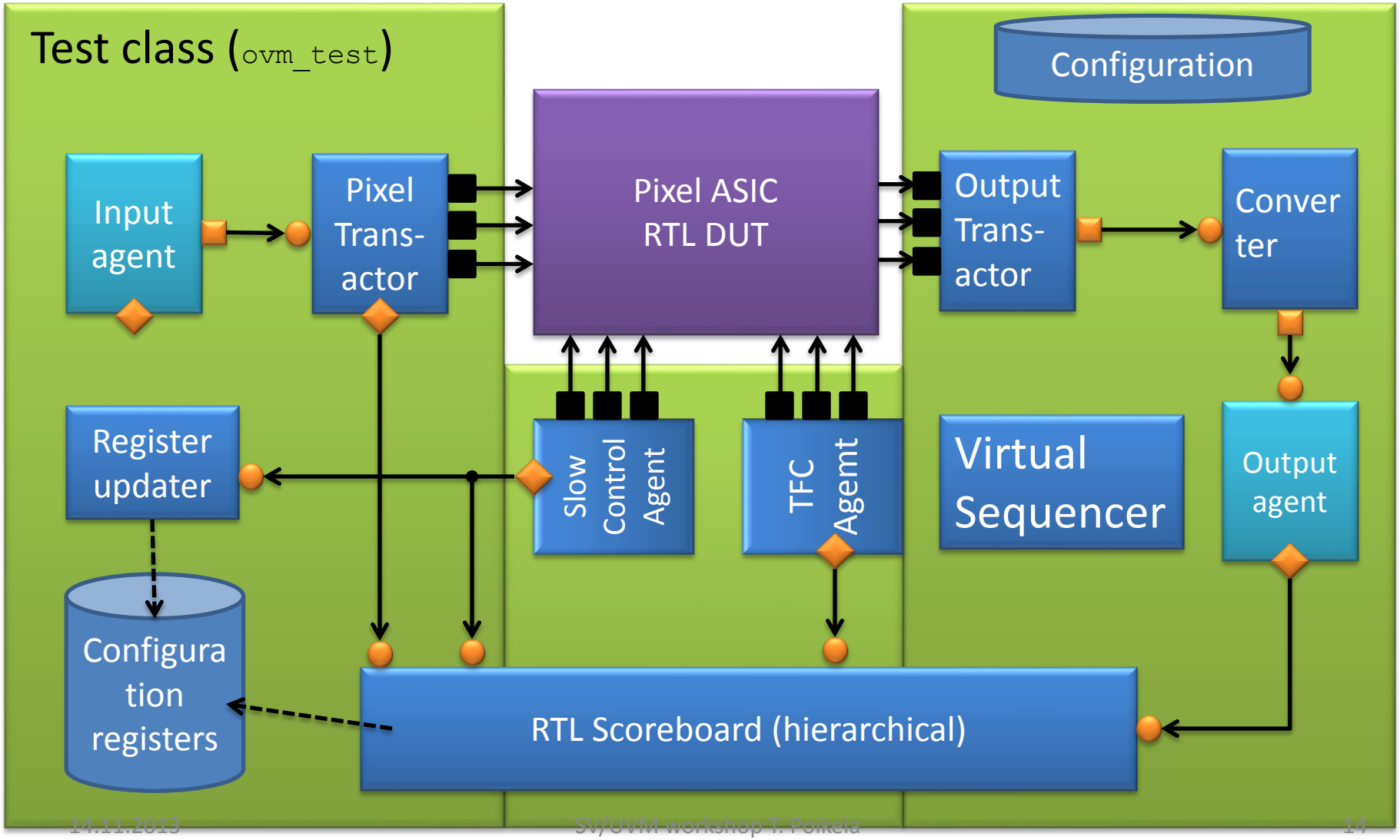
Completely untimed style

- Can use C++ as well
- Seems to be 4-8 x faster than TLM model

Can add timing by using #25ns;
No performance loss.

DESIGN VERIFICATION

RTL Verification Environment



Test base class

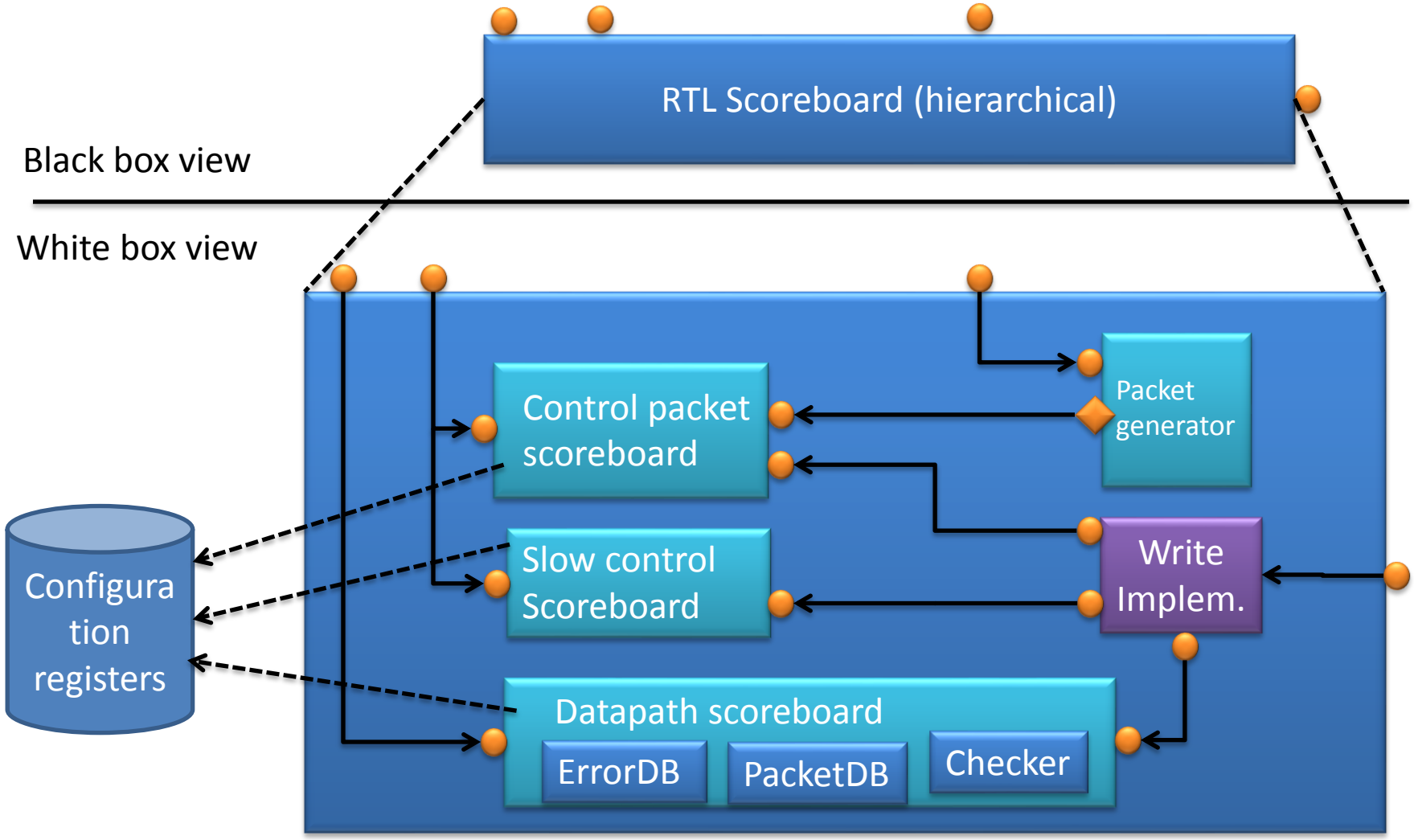
```
class TPX3TopTestBase extends ovm_test;

    `ovm_component_utils(TPX3TopTestBase)

    bit enable_coverage_m           = 1'b0;     ///< If set to 1, coverage is collected.
    bit use_encoded_8b10b_output_m  = 1'b1;     ///< If set to 1, output is 8b/10b encoded
    bit use_timer_m                 = 1'b1;     ///< If set to 1, the on-chip timer is used.
    bit verbose_m                   = 1'b0;     ///< If set to 1, prints info during simulation.

    TPX3ConfRegMap              reg_map_m;        ///< Register/memory map of all configuration
    TPX3TopEnv                  env_m;            ///< Contains the verification environment.
    TPX3TopScoreboard           scoreboard_m;     ///< Top-level scoreboard for output packet ch

    TPX3TopSequencer            v_sequencer_m;    ///< Any sequence in master seq lib can be sta

    TPX2ConfigurationObject     configuration_object_m;   ///< Object for passing parameters int
    TPX2SlowControlConfObject   conf_object_m;

    TPX3TopSeqBase              chip_test_seq_m;  // does not do anything, override this with fa
    TPX3Timer                   timer_m;

    TPX3PixelMaskUpdater    mask_updater_m;

    function new(string name = "chip_rtl_test", ovm_component parent);
        super.new(name, parent);
    endfunction: new
```

Test base class

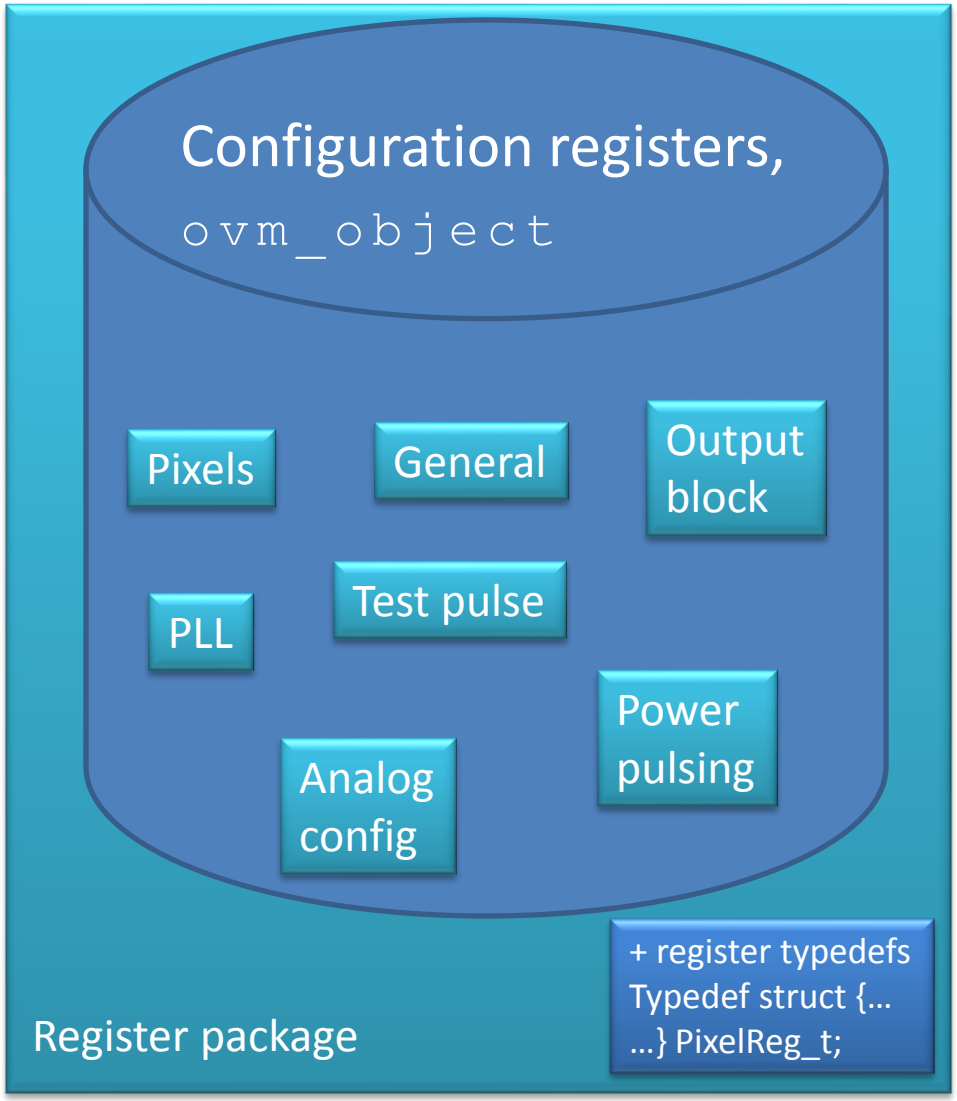
Actual, derived test

```
class TPX3HardResetTest extends TPX3TopTestBase;

    `ovm_component_utils(TPX3HardResetTest)

    function new(string name = "top_reset_test", ovm_component parent);
        super.new(name, parent);
    endfunction: new

    protected virtual function void set_type_overrides();
        super.set_type_overrides();
        set_type_override_by_type(TPX3TopSeqBase::get_type(), // Base ty
            TPX3TopResetSeq::get_type());
    endfunction: set_type_overrides

endclass: TPX3HardResetTest
```

Scoreboard block diagram



Configuration registers



- Created in test class
- Passed to components inside a config object
- Each register is an `ovm_object`, has a coverage model
- Only handful of registers but > 800 LOCs of SV (excluding drivers, macros etc.)

Assertions – using macros

```
Define macro (makes code shorter):
==================================

`define assert_clk( clk_name, my_property )\
    assert property (@(posedge clk_name)\
    disable iff (reset == ACTIVE)\
    my_property )


Usage in RTL code:
==================

/** FIFO must never be full when write signal is high. */
`assert_clk(fifo_wclk, write |-> !full);

/** FIFO must never be empty when read is high. */
`assert_clk(fifo_rclk, read  |-> !empty);
```

Source: C. E. Cummings. SystemVerilog Assertions Design Tricks and SVA Bind Files. 2009.

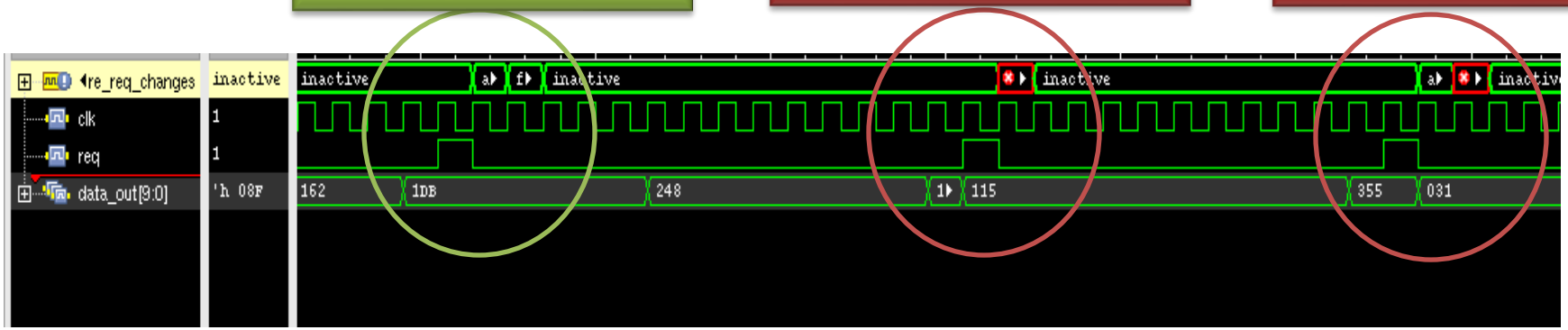
Assertions cont'd

```
Define macro first:
====================
`define prop_sig1_stable_when_sig0_changes(clk_name, rst_name, sig0, sig1)\
    property (\
        @(posedge clk_name)\
        disable iff (reset == ACTIVE)\
        $rose(sig0) |-> (sig1 == $past(sig1)) ##1 (sig1 == $past(sig1))\
    )

Usage in RTL code (inside module, outside always-blocks):
===================
assert `prop_sig1_stable_when_sig0_changes(clk, rst, req, data_out);
```

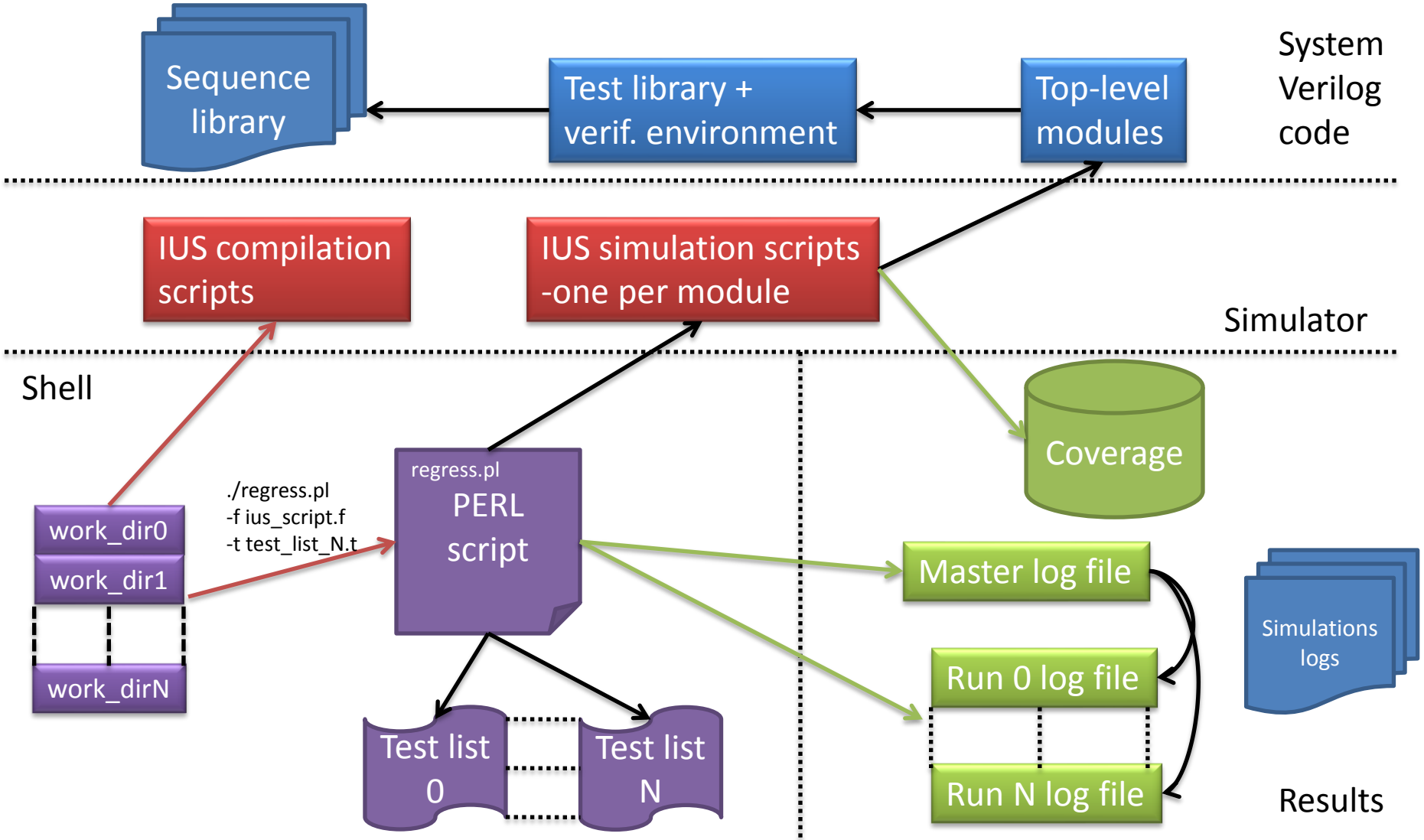
No error. Data stable 1 clock before and after.

ERROR. Data changed 1 clock before req.

ERROR. Data changed 1 clock after req.



Scripting flow



Concluding remarks

- Few use cases of SV + OVM have been presented
  - Modeling of a pixel chip
  - RTL verification: Testbench structure using OVM

- Plan is to move to UVM for Velopix (and maybe change the high-level model from TLM/OVM to pure SV)
- Use assertions!

THANK YOU FOR THE ATTENTION!