

SystemVerilog Assertions Verification with SVAUnit

Ionuț Ciocîrlan
Andra Radu
AMIQ Consulting

June 23, 2015
SNUG UK



Agenda

SystemVerilog Assertions (SVAs)

About SVAUnit

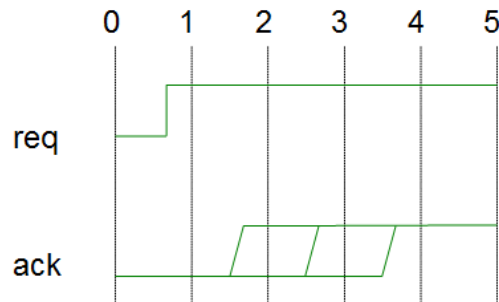
SVAUnit example

SVAUnit Infrastructure

Conclusions

Q&A

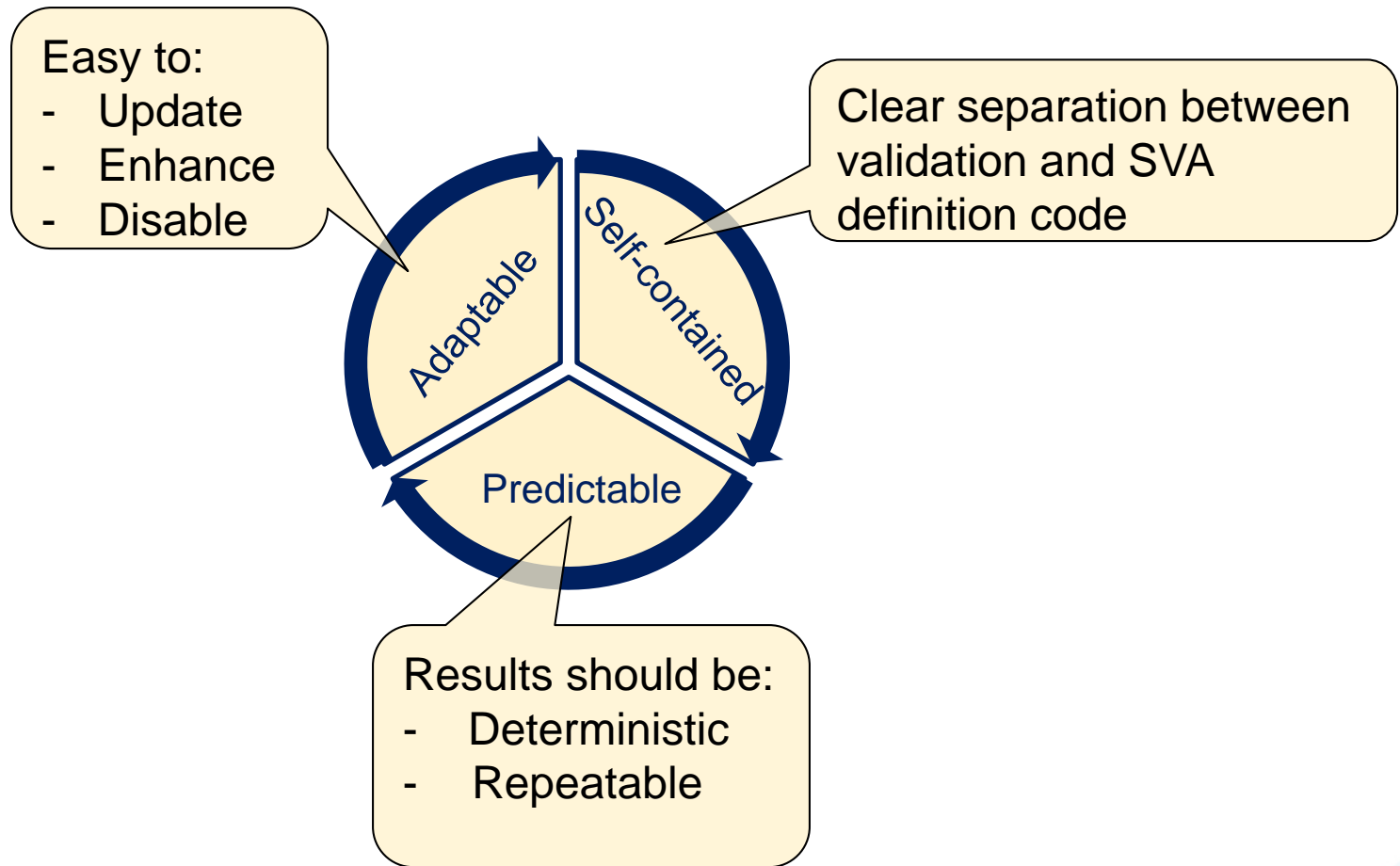
SystemVerilog Assertions (SVAs)



After the rise of request signal, the acknowledge signal should be asserted no later than 3 clocks cycles.

- What is an assertion?
 - A check against the specification of a design that it never violates.
- Why use SVA?
 - Powerful feature, flexible, measurable and with a concise syntax.

SVA Verification Challenges

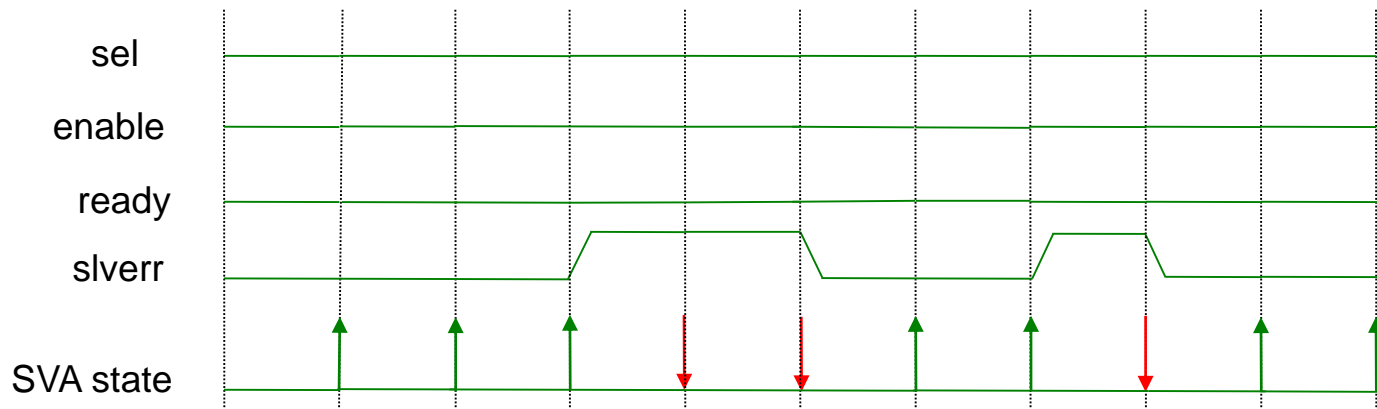


Introducing SVAUnit

- Structured framework for Unit Testing for SVAs
- Allows the user to decouple the SVA definition from its validation code
- UVM compliant package written in SystemVerilog
- Encapsulate each SVA testing scenario inside an unit test
- Easily controlled and supervised using a simple API

Hands-on example – What to check

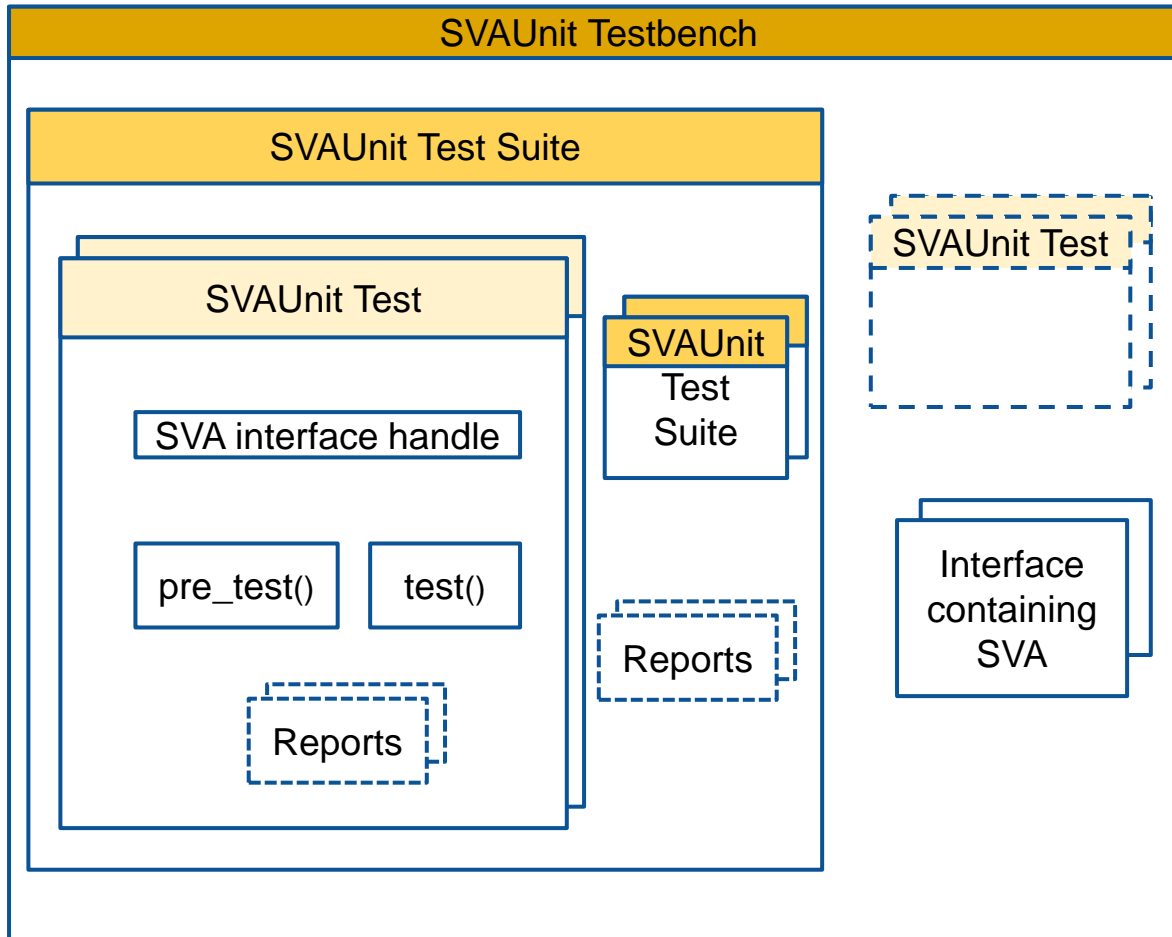
“slverr signal should be 0 if no slave is selected or when transfer is not enabled or when slave is not ready to respond”



```
interface an_if (input clk);  
  property an_sva_property;  
    @(posedge clk)  
    !sel or !enable or !ready |-> !slverr;  
  endproperty  
  AN_SVA: assert property (an_sva_property) else  
    `uvm_error("AN_SVA", "AN_SVA failed")  
endinterface
```

↑ - SVA succeeded
↓ - SVA failed

SVAUnit Environment Architecture



- **SVAUnit Testbench**
 - Enables SVAUnit
 - Instantiates SVA interface
 - Starts test
- **SVAUnit Test**
 - Contains the SVA scenario
- **SVAUnit Test Suite**
 - Test and test suite container

Example of SVAUnit Testbench

```

module top;
  // Instantiate the SVAUnit framework
  `SVAUNIT_UTILS
  ...

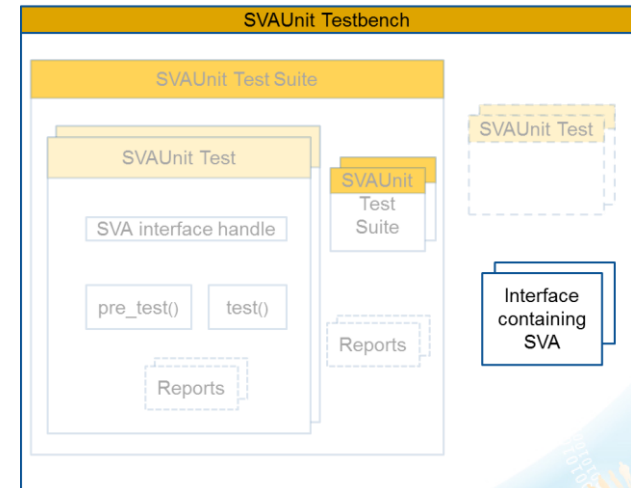
  // Instantiate the interface with the SVAs we want to test
  an_if dut_if(.clk(clock));

  initial begin
    // Register the interface with the uvm_config_db
    uvm_config_db#(virtual an_if)::
      set(uvm_root::get(), "*", "VIF", dut_if);

    // Start the scenarios
    run_test();
  end

  ...
endmodule

```



Example of SVAUnit Test

```

class ut1 extends svaunit_test;
  // The virtual interface used to drive the signals
  virtual an_if vif;

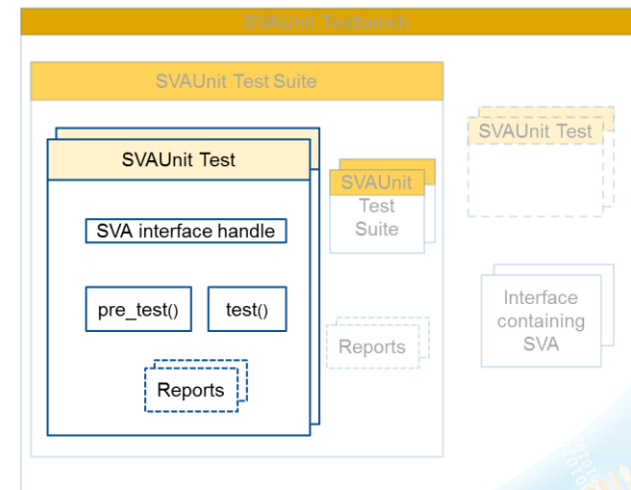
  function void build_phase(input uvm_phase phase);
    // Retrieve the interface handle from the uvm_config_db
    if (!uvm_config_db#(virtual an_if)::get(this, "", "vif", vif))
      `uvm_fatal("UT1_NO_VIF_ERR", "SVA interface is not set!")

    // Test will run by default;
    disable_test();
  endfunction

  task pre_test();
    // Initialize signals
  endtask

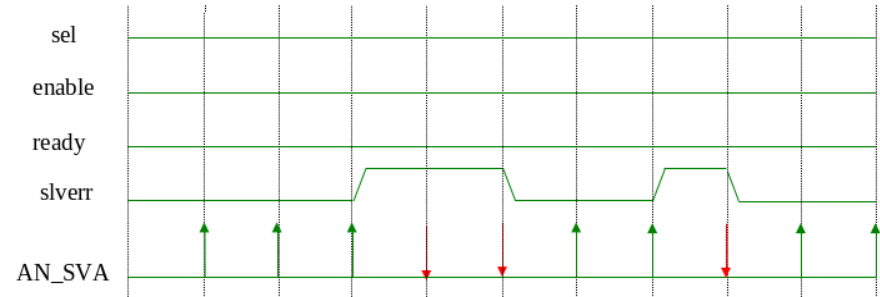
  task test();
    // Create scenarios for AN_SVA
  endtask
endclass

```



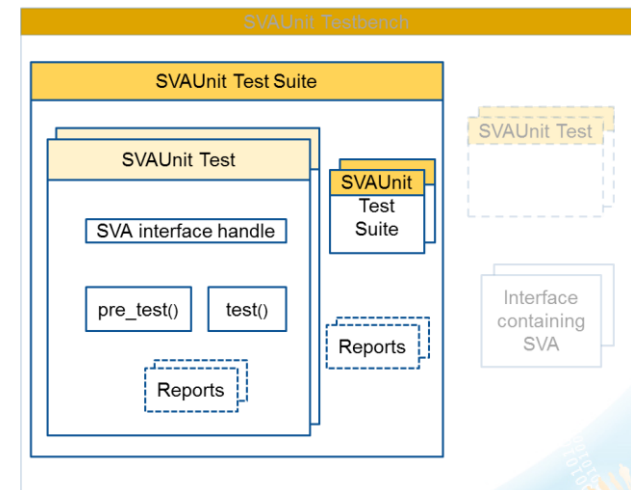
Example of test() task

```
task test();  
    // Create scenarios for AN_SVA  
    disable_all_assertions();  
    enable_assertion("AN_SVA");  
  
    repeat(3) begin  
        @(posedge vif.clk);  
        fail_if_sva_not_succeeded("AN_SVA", "The assertion should have  
succeeded");  
    end  
  
    // Trigger the error scenario  
    vif.slverr = 1'b1;  
  
    repeat(2) begin  
        @(posedge vif.clk);  
        fail_if_sva_succeeded("AN_SVA", "The assertion should have failed");  
    end  
  
    vif.slverr = 1'b0;  
    ...  
endtask
```



Example of SVAUnit Test Suite

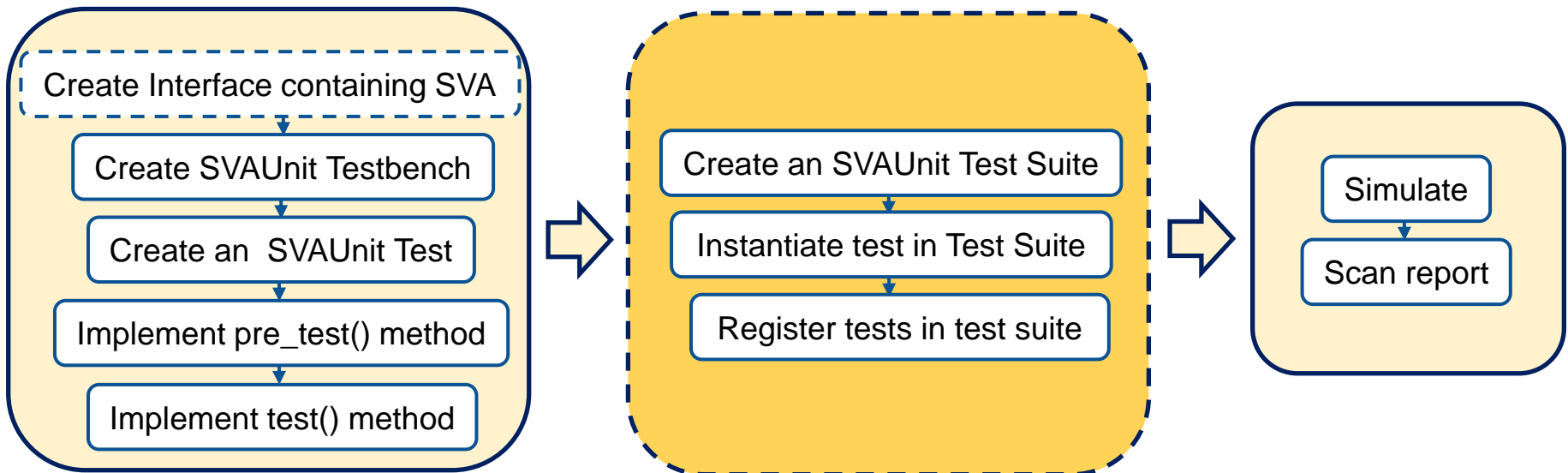
```
class uts extends svaunit_test_suite;  
  // Instantiate the SVAUnit tests  
  ut1 ut1;  
  ...  
  ut10 ut10;  
  
  function void build_phase(input uvm_phase phase);  
    ut1 = ut1::type_id::create("ut1", this);  
    ...  
    ut10 = ut10::type_id::create("ut10", this);  
  
    // Register tests in suite  
    add_test(ut1);  
    ...  
    add_test(ut10);  
  endfunction  
  
endclass
```



SVAUnit Test API

Control	<ul style="list-style-type: none">• <code>disable_all_assertions();</code>• <code>enable_assertion(sva_name);</code>• <code>enable_all_assertions();</code>• ...
Check	<ul style="list-style-type: none">• <code>fail_if_sva_does_not_exists(sva_name, error_msg);</code>• <code>pass_if_sva_not_succeeded(sva_name, error_msg);</code>• <code>pass/fail_if(expression, error_msg);</code>• ...
Report	<ul style="list-style-type: none">• <code>print_status();</code>• <code>print_sva();</code>• <code>print_report();</code>• ...

SVAUnit Flow



Error reporting

Name of SVAUnit
check

SVAUnit test path

```
UVM_ERROR @ 55000 ns [SVAUNIT_FAIL_IF_SVA_SUCCEEDED_ERR]: [x_z_suite.addr_x_z_test::x_z_addr_ut  
AMIQ_APB_ILLEGAL_ADDR_VALUE_ERR] The assertion should have failed
```

Name of SVA under
test

Custom error
message

Hierarchy report

```
UVM_INFO @ 56000 ns [protocol_ts]:  
  protocol_ts  
    protocol_ts.protocol_test1  
    protocol_ts.protocol_test2  
    protocol_ts.x_z_suite  
      x_z_suite.addr_x_z_test  
      x_z_suite.slvrr_x_z_test  
      x_z_suite.sel_x_z_test  
      x_z_suite.write_x_z_test  
      x_z_suite.strb_x_z_test  
      x_z_suite.prot_x_z_test  
      x_z_suite.enable_x_z_test  
      x_z_suite.ready_x_z_test
```

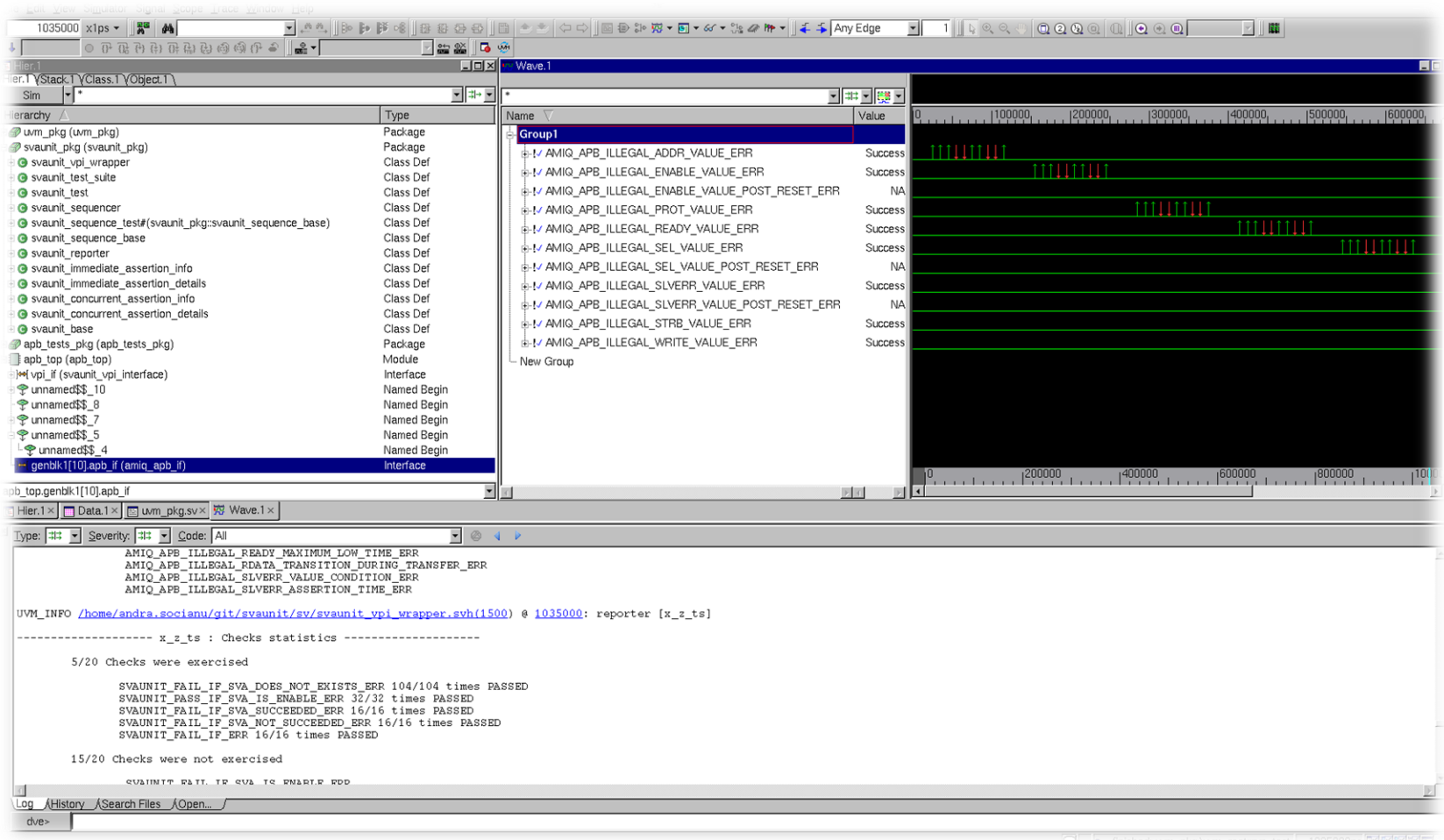
Test scenarios exercised

```
----- protocol_ts test suite : Status statistics -----  
  
* protocol_ts FAIL (2/3 test cases PASSED)  
  * protocol_ts.x_z_suite FAIL (0/8 test cases PASSED)  
    protocol_ts.protocol_test2 PASS (13/13 assertions PASSED)  
    protocol_ts.protocol_test1 PASS (13/13 assertions PASSED)  
  
UVM_INFO @ 56000 ns [protocol_ts]:  
  
    3/3 Tests ran during simulation  
  
        protocol_ts.x_z_suite  
        protocol_ts.protocol_test2  
        protocol_ts.protocol_test1
```


SVAs and checks exercised

```
----- protocol_ts test suite : SVA and checks statistics -----  
  
  AMIQ_APB_ILLEGAL_SEL_TRANSITION_TR_PHASES_ERR  13/13 checks PASSED  
    SVAUNIT_FAIL_IF_SVA_SUCCEEDED_ERR 1/1 times PASSED  
    SVAUNIT_FAIL_IF_SVA_NOT_SUCCEEDED_ERR 2/2 times PASSED  
    SVAUNIT_FAIL_IF_SVA_DOES_NOT_EXISTS_ERR 7/7 times PASSED  
    SVAUNIT_PASS_IF_SVA_IS_ENABLE_ERR 3/3 times PASSED  
  
  AMIQ_APB_ILLEGAL_SEL_TRANSITION_DURING_TRANSFER_ERR  13/13 checks PASSED  
    SVAUNIT_FAIL_IF_SVA_NOT_SUCCEEDED_ERR 1/1 times PASSED  
    SVAUNIT_FAIL_IF_SVA_SUCCEEDED_ERR 2/2 times PASSED  
    SVAUNIT_FAIL_IF_SVA_DOES_NOT_EXISTS_ERR 7/7 times PASSED  
    SVAUNIT_PASS_IF_SVA_IS_ENABLE_ERR 3/3 times PASSED
```

Tools integration



The screenshot displays a simulation tool interface with three main panels:

- Hierarchy Panel (Left):** Shows a tree structure of test classes and packages. The selected item is `genblk1[10].apb_if (amiq_apb_if)`.
- Name/Value Panel (Middle):** Lists various error types and their status.

Name	Value
Group1	Success
AMIQ_APB_ILLEGAL_ADDR_VALUE_ERR	Success
AMIQ_APB_ILLEGAL_ENABLE_VALUE_ERR	Success
AMIQ_APB_ILLEGAL_ENABLE_VALUE_POST_RESET_ERR	NA
AMIQ_APB_ILLEGAL_PROT_VALUE_ERR	Success
AMIQ_APB_ILLEGAL_READY_VALUE_ERR	Success
AMIQ_APB_ILLEGAL_SEL_VALUE_ERR	Success
AMIQ_APB_ILLEGAL_SEL_VALUE_POST_RESET_ERR	NA
AMIQ_APB_ILLEGAL_SLVERR_VALUE_ERR	Success
AMIQ_APB_ILLEGAL_SLVERR_VALUE_POST_RESET_ERR	NA
AMIQ_APB_ILLEGAL_STRB_VALUE_ERR	Success
AMIQ_APB_ILLEGAL_WRITE_VALUE_ERR	Success
New Group	
- Waveform Viewer (Right):** Shows a timing diagram with multiple signals plotted against time (0 to 1,600,000).

At the bottom, a console window displays the following text:

```

Type: # Severity: # Code: All
AMIQ_APB_ILLEGAL_READY_MAXIMUM_LOW_TIME_ERR
AMIQ_APB_ILLEGAL_RDATA_TRANSITION_DURING_TRANSFER_ERR
AMIQ_APB_ILLEGAL_SLVERR_VALUE_CONDITION_ERR
AMIQ_APB_ILLEGAL_SLVERR_ASSERTION_TIME_ERR

UVM_INFO /home/andra.socianu/git/svaunit/sv/svaunit_vpi_wrapper.svh(1500) @ 1035000: reporter [x_z_ts]

----- x_z_ts : Checks statistics -----
5/20 Checks were exercised

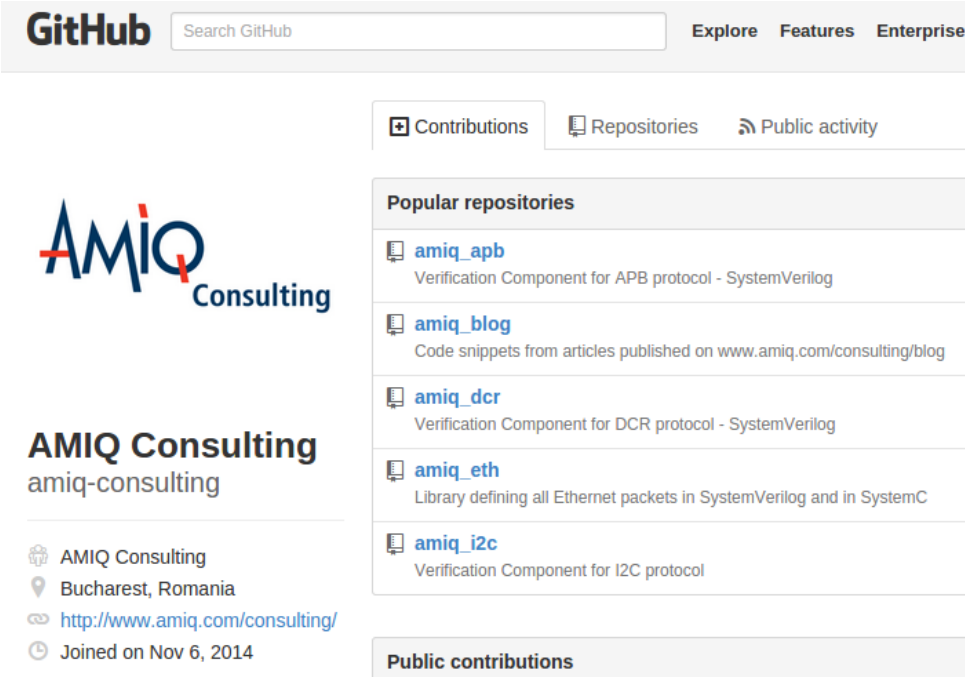
SVAUNIT_FAIL_IF_SVA_DOES_NOT_EXISTS_ERR 104/104 times PASSED
SVAUNIT_PASS_IF_SVA_IS_ENABLE_ERR 32/32 times PASSED
SVAUNIT_FAIL_IF_SVA_SUCCEEDED_ERR 16/16 times PASSED
SVAUNIT_FAIL_IF_SVA_NOT_SUCCEEDED_ERR 16/16 times PASSED
SVAUNIT_FAIL_IF_ERR 16/16 times PASSED

15/20 Checks were not exercised
SVAUNIT_FAIL_IF_SVA_IS_ENABLE_ERR
    
```

Conclusions

- SVAUnit decouples the checking logic from SVA definition code
- Safety net for eventual code refactoring
- Can also be used as self-checking documentation on how SVAs work
- Quick learning curve
- Easy-to-use and flexible API
- Speeds up verification closure
- Boosts verification quality

Availability



The screenshot shows the GitHub profile page for AMIQ Consulting. At the top, there is a search bar and navigation links for 'Explore', 'Features', and 'Enterprise'. Below the search bar are tabs for 'Contributions', 'Repositories', and 'Public activity'. The profile name 'AMIQ Consulting' is displayed with a logo. The repository list includes:

- amiq_apb**: Verification Component for APB protocol - SystemVerilog
- amiq_blog**: Code snippets from articles published on www.amiq.com/consulting/blog
- amiq_dcr**: Verification Component for DCR protocol - SystemVerilog
- amiq_eth**: Library defining all Ethernet packets in SystemVerilog and in SystemC
- amiq_i2c**: Verification Component for I2C protocol

Below the repository list, there is a section for 'Public contributions'.

- SVAUnit is an open-source package released by AMIQ Consulting
- We provide:
 - SystemVerilog and simulator integration codes
 - AMBA-APB assertion package
 - Code templates and examples
 - HTML documentation for API

<https://github.com/amiq-consulting/svaunit>

Q & A



Thank You

