2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
DECEMBER 6 - 7, 2022

How creativity kills reuse – A modern take on UVM/SV TB architectures

Andrei Vintila, Sergiu Duda

AMIQ Consulting

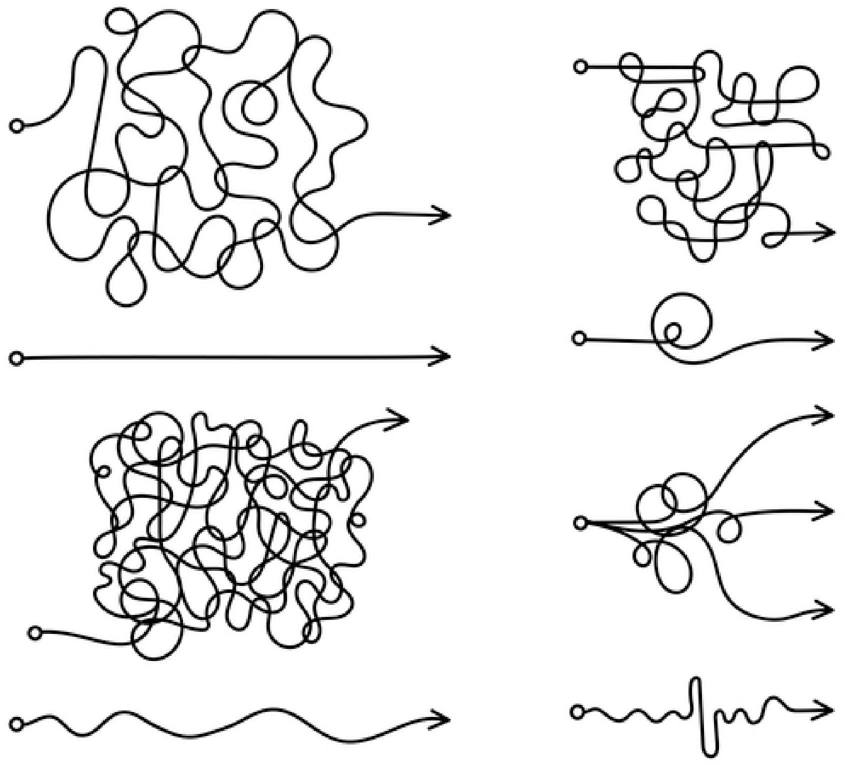
AMIQ Consulting

accellera SYSTEMS INITIATIVE

Contents

Motivation

- Why change it if it works?
- Complexity vs Functionality
- Backend vs Frontend TB development
- Divide and conquer



Scope

- Architecture and Guidelines
- Founding blocks
- Proof of concept and building a TB
- Automation and future work

Concept and Rules

- Environment should be organized in such a way that all and any component and object can exist without a dependency on the other (Type agnostic / Allows runtime scaling up/down)
- All sequences should be organized as stand-alone entities, that can dynamically form a testcase based on their order and constraints
- All control variables, as well as stimuli relevant variables, should be registered as plusargs
- Creating a testcase means picking an environment and a collection of sequences

Plusargs and text inputs

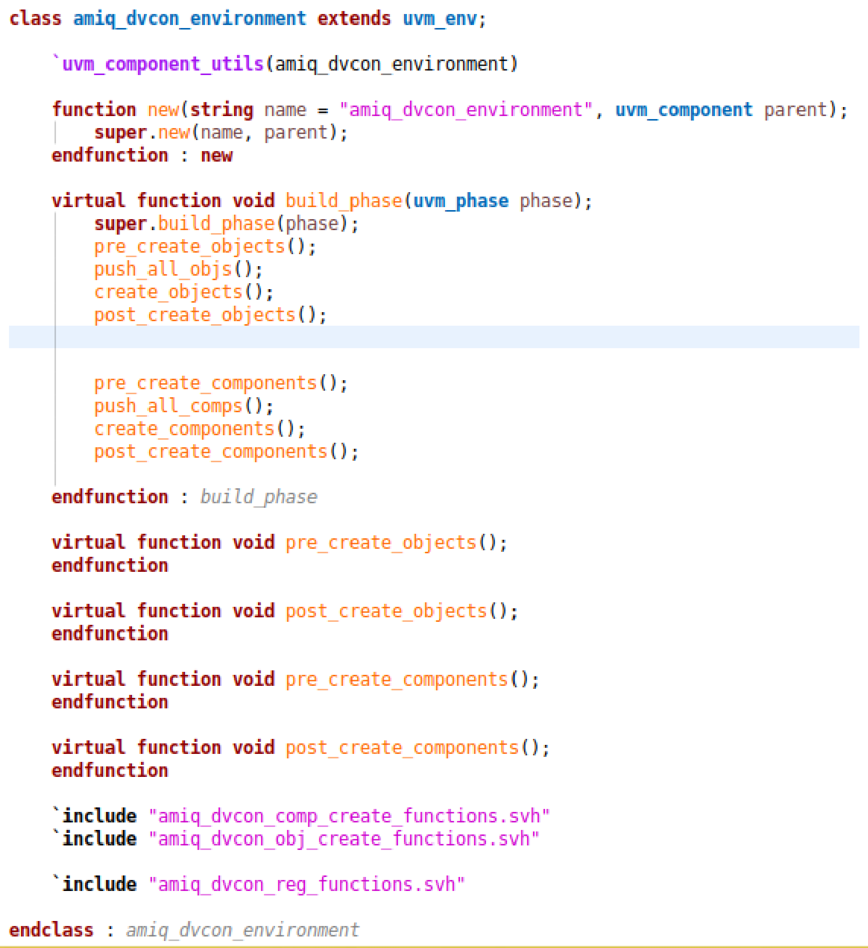
- Bare plusargs vs UVM set config var
- Scalability and speed
- Text file control
- Control from outside the sim

```
+env_cfg_0_vip0_is_active=1
+env_cfg_0_vip0_has_checks=1
+env_cfg_0_vip0_has_has_coverage=0
+env_cfg_0_vip3_is_active=1
+env_cfg_0_vip3_has_checks=1
+env_cfg_0_vip3_has_has_coverage=0
+env_cfg_0_vip5_is_active=0
+env_cfg_0_vip5_has_checks=1
+env_cfg_0_vip5_has_has_coverage=1
```

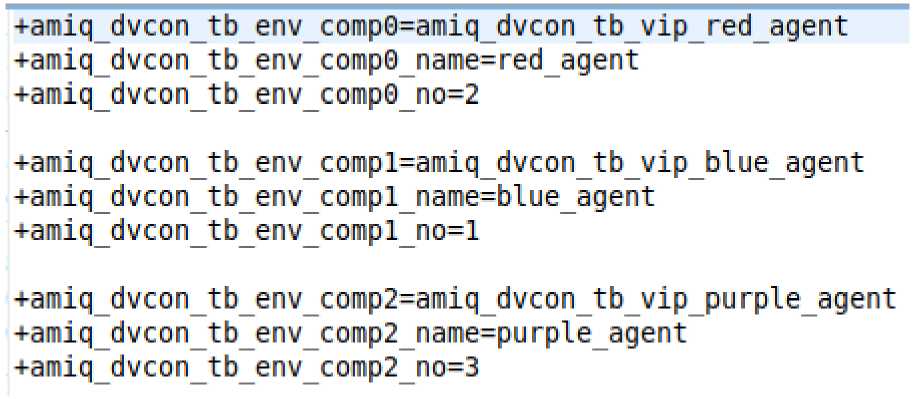
Environment setup (1)

```
uvm_component components[$];
string component_types[$];
string component_names[$];

function void create_components();
    uvm_factory factory; // Handle to the UVM factory that allows the creation of any object via string
    // Variable used for creating multiple components of the same type
    int cnt;

    // Retrieve the factory
    factory = uvm_factory::get();

    // Creating all of the available components
    foreach(component_types[i]) begin
        uvm_component local_component; // Handle to the current uvm component that is going to be created this 'foreach' iteration

        // Create component
        local_component = factory.create_component_by_name(component_types[i], this.get_full_name(), component_names[i], this);
        components.push_back(local_component);
    end
endfunction

function void push_all_comps();
    int index;
    string comp_type;
    string comp_name;
    string comp_name_with_index;
    int comp_number;

    forever begin

        // Retrieve the components passed as plusargs
        comp_type = retrieve_comp_type(index);

        // If no component name is retrieved, we break the loop
        if(comp_type == "") break;

        // Retrieve the number of agents for the current comp type
        // and add them to the queue
        comp_number = retrieve_comp_number(index);

        // Add the component types to the queue according to the "number of" set
        repeat(comp_number)
            component_types.push_back(comp_type);

        // Retrieve the name of the agent
        comp_name = retrieve_comp_name(index);

        // If the component name is not defined, set it to the type name, otherwise use the component name given
        // If there are multiple instances defined on this comp index, then create all of them incrementally
        // adding incremental "_*index*" at the end
        if(comp_name == "")
            for(int i=0; i<comp_number;i++)
                component_names.push_back(unique_name_check(comp_type, i, (comp_number==1)));
        else
            for(int i=0; i<comp_number;i++)
                component_names.push_back(unique_name_check(comp_name, i, (comp_number==1)));

        // Go to the next index
        index++;
    end

endfunction

function string retrieve_comp_type(int index);
    string name_comp_index = $sformatf("%0s_comp%0d", get_name(), index);
    `uvm_info(get_name(), $sformatf("Looking for comp %s", name_comp_index), UVM_NONE)
    if(!$value$plusargs({name_comp_index, "=%0s"}, retrieve_comp_type)) retrieve_comp_type = "";
endfunction

function int retrieve_comp_number(int index);
    string name_comp_index_no = $sformatf("%0s_comp%0d_no", get_name(), index);
    `uvm_info(get_name(), $sformatf("Looking for no %s", name_comp_index_no), UVM_NONE)
    if(!$value$plusargs({name_comp_index_no, "=%0d"}, retrieve_comp_number)) retrieve_comp_number = 1;
endfunction

function string retrieve_comp_name(int index);
    string name_comp_index_name = $sformatf("%0s_comp%0d_name", get_name(), index);
    `uvm_info(get_name(), $sformatf("Looking for name %s", name_comp_index_name), UVM_NONE)
    if(!$value$plusargs({name_comp_index_name, "=%0s"}, retrieve_comp_name)) retrieve_comp_name = "";
endfunction
```

Environment setup (2)



```
class amiq_dvcon_environment extends uvm_env;

    `uvm_component_utils(amiq_dvcon_environment)

    function new(string name = "amiq_dvcon_environment", uvm_component parent);
        super.new(name, parent);
    endfunction : new

    virtual function void build_phase(uvm_phase phase);
        super.build_phase(phase);
        pre_create_objects();
        push_all_objs();
        create_objects();
        post_create_objects();


        pre_create_components();
        push_all_comps();
        create_components();
        post_create_components();
    endfunction : build_phase

    virtual function void pre_create_objects();
    endfunction

    virtual function void post_create_objects();
    endfunction

    virtual function void pre_create_components();
    endfunction

    virtual function void post_create_components();
    endfunction

    `include "amiq_dvcon_comp_create_functions.svh"
    `include "amiq_dvcon_obj_create_functions.svh"

    `include "amiq_dvcon_reg_functions.svh"

endclass : amiq_dvcon_environment
```



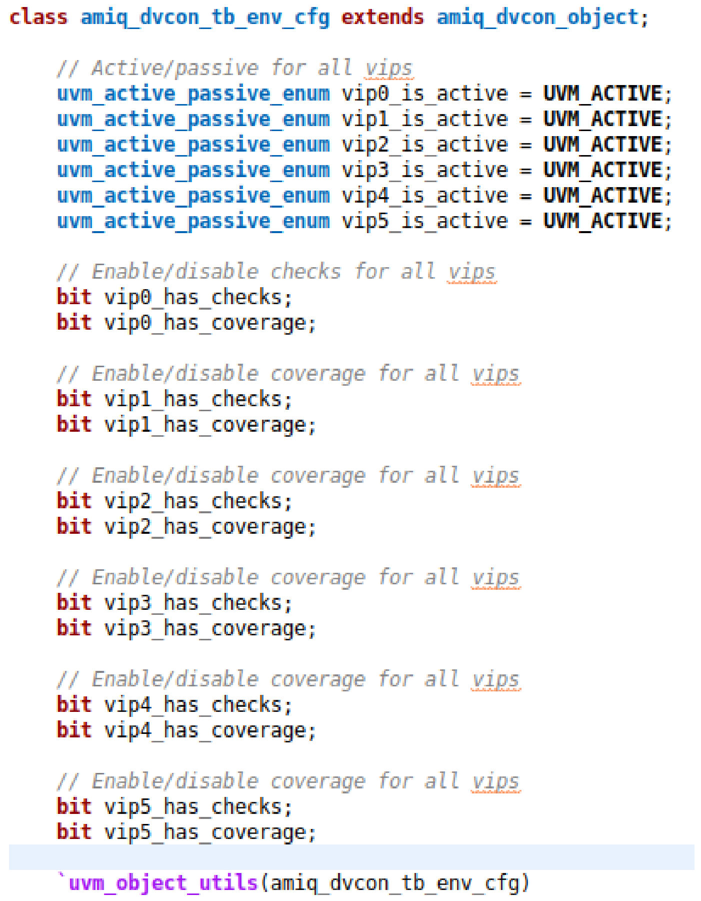
```
+amiq_dvcon_tb_env_comp0=amiq_dvcon_tb_vip_red_agent
+amiq_dvcon_tb_env_comp0_name=red_agent
+amiq_dvcon_tb_env_comp0_no=2

+amiq_dvcon_tb_env_comp1=amiq_dvcon_tb_vip_blue_agent
+amiq_dvcon_tb_env_comp1_name=blue_agent
+amiq_dvcon_tb_env_comp1_no=1

+amiq_dvcon_tb_env_comp2=amiq_dvcon_tb_vip_purple_agent
+amiq_dvcon_tb_env_comp2_name=purple_agent
+amiq_dvcon_tb_env_comp2_no=3
```

Environment Setup (3)

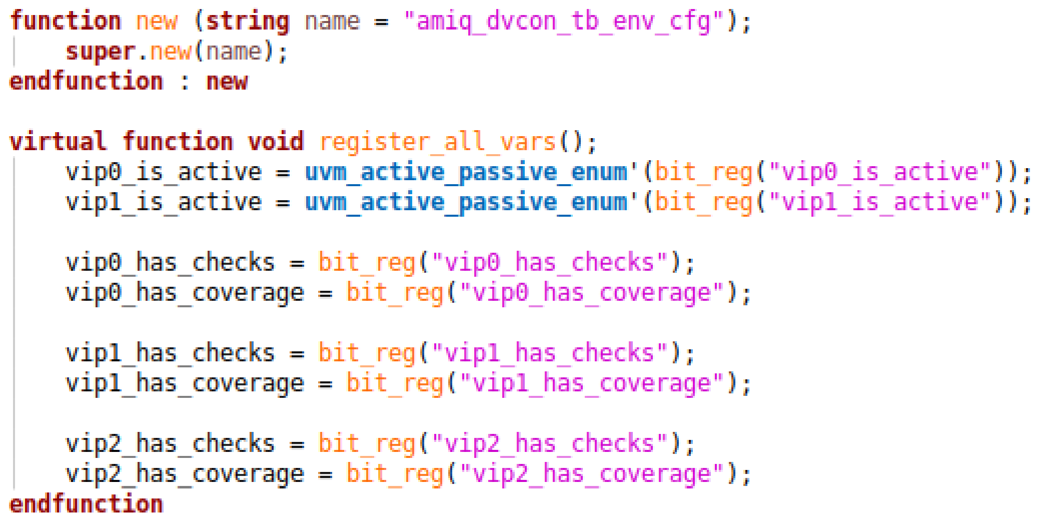
```
class amiq_dvcon_object extends uvm_object;

    `uvm_object_utils(amiq_dvcon_object)

    function new(string name="");
        super.new(name);
        register_all_vars();
    endfunction

    `include "amiq_dvcon_reg_functions.svh"

endclass
```

```
+amiq_dvcon_tb_env_obj0=amiq_dvcon_tb_env_cfg
+amiq_dvcon_tb_env_obj0_name=env_cfg_0
+env_cfg_0_vip0_is_active=1
+env_cfg_0_vip0_has_checks=1
+env_cfg_0_vip0_has_has_coverage=0
+env_cfg_0_vip3_is_active=1
+env_cfg_0_vip3_has_checks=1
+env_cfg_0_vip3_has_has_coverage=0
+env_cfg_0_vip5_is_active=0
+env_cfg_0_vip5_has_checks=1
+env_cfg_0_vip5_has_has_coverage=1
```

Environment Setup (4)

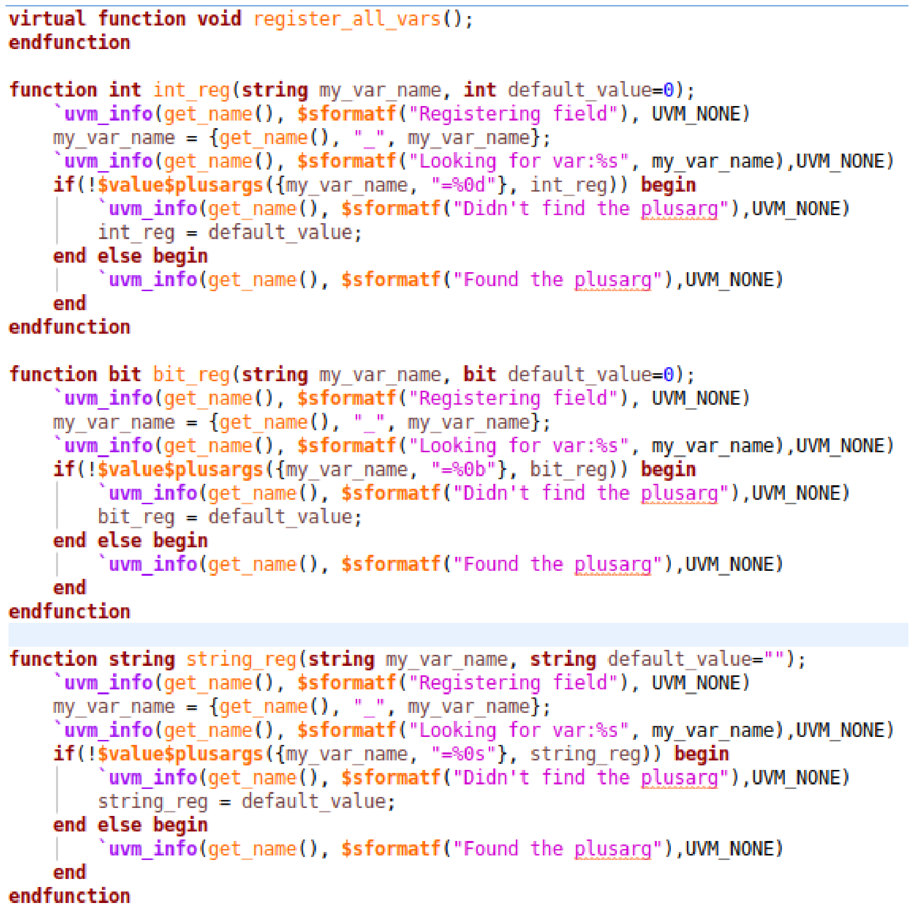


```
class amiq_dvcon_tb_env_cfg extends amiq_dvcon_object;

    // Active/passive for all vips
    uvm_active_passive_enum vip0_is_active = UVM_ACTIVE;
    uvm_active_passive_enum vip1_is_active = UVM_ACTIVE;
    uvm_active_passive_enum vip2_is_active = UVM_ACTIVE;
    uvm_active_passive_enum vip3_is_active = UVM_ACTIVE;
    uvm_active_passive_enum vip4_is_active = UVM_ACTIVE;
    uvm_active_passive_enum vip5_is_active = UVM_ACTIVE;

    // Enable/disable checks for all vips
    bit vip0_has_checks;
    bit vip0_has_coverage;

    // Enable/disable coverage for all vips
    bit vip1_has_checks;
    bit vip1_has_coverage;

    // Enable/disable coverage for all vips
    bit vip2_has_checks;
    bit vip2_has_coverage;

    // Enable/disable coverage for all vips
    bit vip3_has_checks;
    bit vip3_has_coverage;

    // Enable/disable coverage for all vips
    bit vip4_has_checks;
    bit vip4_has_coverage;

    // Enable/disable coverage for all vips
    bit vip5_has_checks;
    bit vip5_has_coverage;

    `uvm_object_utils(amiq_dvcon_tb_env_cfg)
```

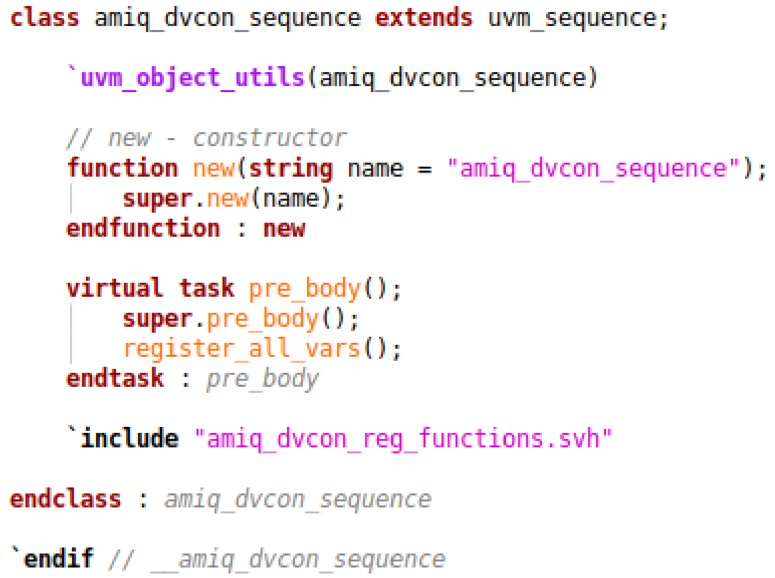


```
function new (string name = "amiq_dvcon_tb_env_cfg");
    super.new(name);
endfunction : new

virtual function void register_all_vars();
    vip0_is_active = uvm_active_passive_enum'(bit_reg("vip0_is_active"));
    vip1_is_active = uvm_active_passive_enum'(bit_reg("vip1_is_active"));

    vip0_has_checks = bit_reg("vip0_has_checks");
    vip0_has_coverage = bit_reg("vip0_has_coverage");

    vip1_has_checks = bit_reg("vip1_has_checks");
    vip1_has_coverage = bit_reg("vip1_has_coverage");

    vip2_has_checks = bit_reg("vip2_has_checks");
    vip2_has_coverage = bit_reg("vip2_has_coverage");
endfunction
```

Sequence control (1)

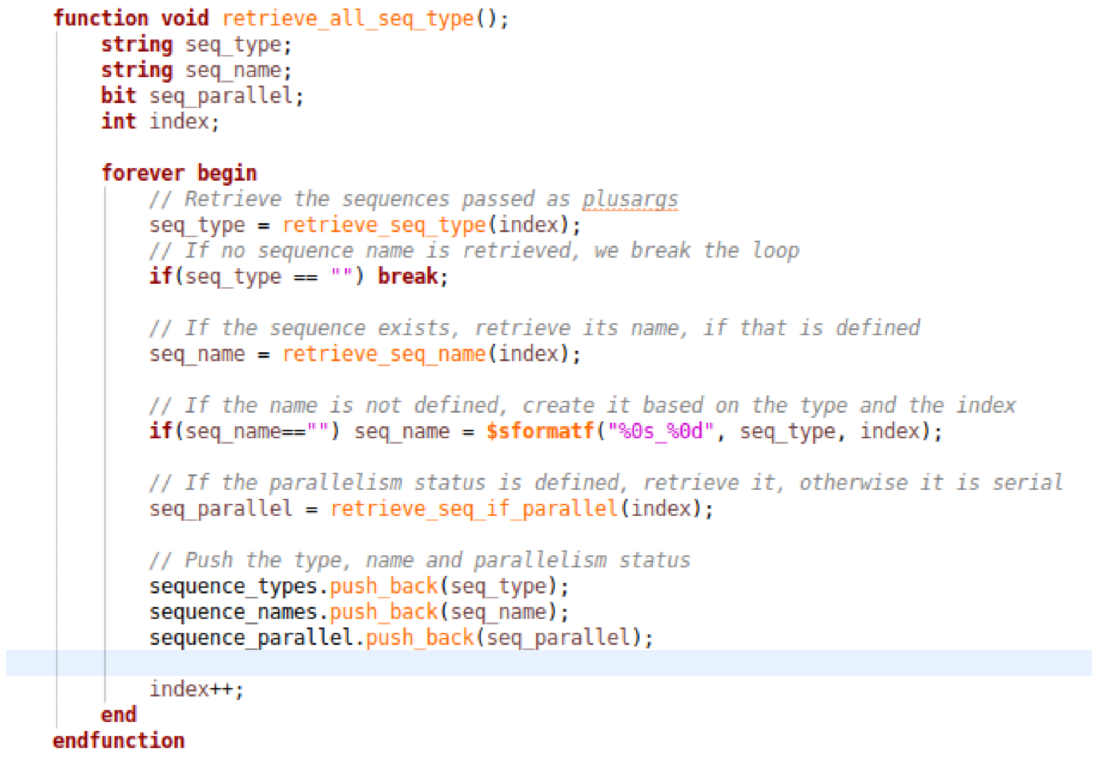


```
virtual function void register_all_vars();
endfunction

function int int_reg(string my_var_name, int default_value=0);
    `uvm_info(get_name(), $sformatf("Registering field"), UVM_NONE)
    my_var_name = {get_name(), "_", my_var_name};
    `uvm_info(get_name(), $sformatf("Looking for var:%s", my_var_name),UVM_NONE)
    if(!$value$plusargs({my_var_name, "=%0d"}, int_reg)) begin
        `uvm_info(get_name(), $sformatf("Didn't find the plusarg"),UVM_NONE)
        int_reg = default_value;
    end else begin
        `uvm_info(get_name(), $sformatf("Found the plusarg"),UVM_NONE)
    end
endfunction

function bit bit_reg(string my_var_name, bit default_value=0);
    `uvm_info(get_name(), $sformatf("Registering field"), UVM_NONE)
    my_var_name = {get_name(), "_", my_var_name};
    `uvm_info(get_name(), $sformatf("Looking for var:%s", my_var_name),UVM_NONE)
    if(!$value$plusargs({my_var_name, "=%0b"}, bit_reg)) begin
        `uvm_info(get_name(), $sformatf("Didn't find the plusarg"),UVM_NONE)
        bit_reg = default_value;
    end else begin
        `uvm_info(get_name(), $sformatf("Found the plusarg"),UVM_NONE)
    end
endfunction

function string string_reg(string my_var_name, string default_value="");
    `uvm_info(get_name(), $sformatf("Registering field"), UVM_NONE)
    my_var_name = {get_name(), "_", my_var_name};
    `uvm_info(get_name(), $sformatf("Looking for var:%s", my_var_name),UVM_NONE)
    if(!$value$plusargs({my_var_name, "=%0s"}, string_reg)) begin
        `uvm_info(get_name(), $sformatf("Didn't find the plusarg"),UVM_NONE)
        string_reg = default_value;
    end else begin
        `uvm_info(get_name(), $sformatf("Found the plusarg"),UVM_NONE)
    end
endfunction
```

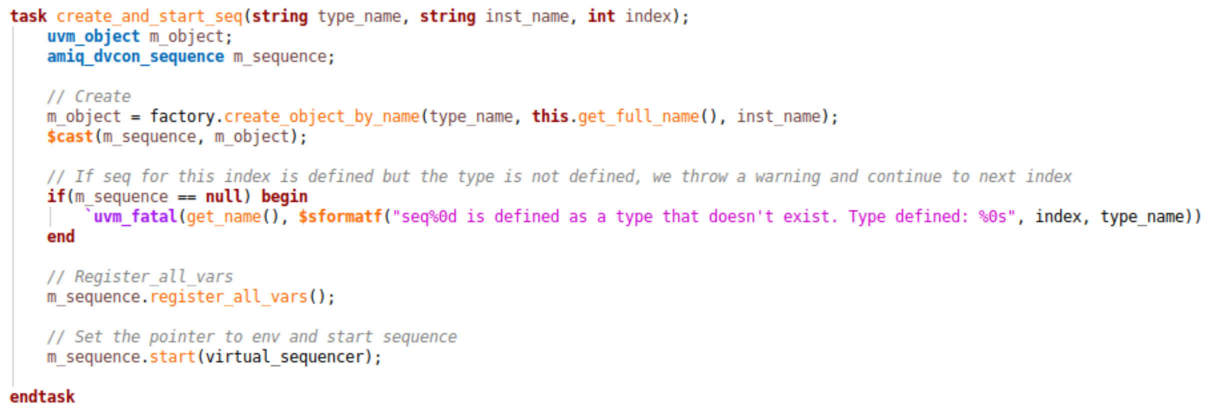


```
class amiq_dvcon_sequence extends uvm_sequence;

    `uvm_object_utils(amiq_dvcon_sequence)

    // new - constructor
    function new(string name = "amiq_dvcon_sequence");
        super.new(name);
    endfunction : new

    virtual task pre_body();
        super.pre_body();
        register_all_vars();
    endtask : pre_body

    `include "amiq_dvcon_reg_functions.svh"

endclass : amiq_dvcon_sequence

`endif // __amiq_dvcon_sequence
```

Sequence control (2)

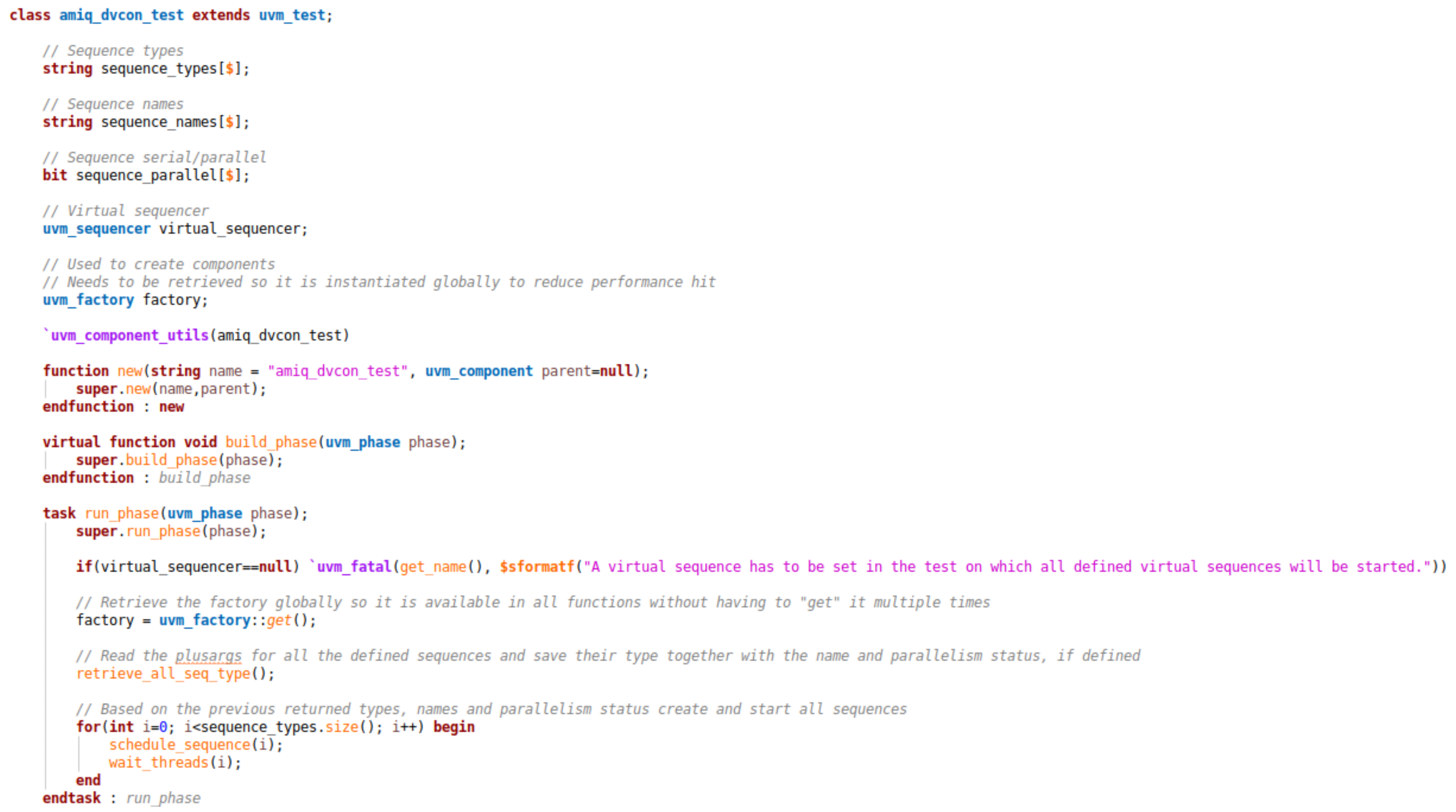


```
function void retrieve_all_seq_type();
    string seq_type;
    string seq_name;
    bit seq_parallel;
    int index;

    forever begin
        // Retrieve the sequences passed as plusargs
        seq_type = retrieve_seq_type(index);
        // If no sequence name is retrieved, we break the loop
        if(seq_type == "") break;

        // If the sequence exists, retrieve its name, if that is defined
        seq_name = retrieve_seq_name(index);

        // If the name is not defined, create it based on the type and the index
        if(seq_name=="") seq_name = $sformatf("%0s_%0d", seq_type, index);

        // If the parallelism status is defined, retrieve it, otherwise it is serial
        seq_parallel = retrieve_seq_if_parallel(index);

        // Push the type, name and parallelism status
        sequence_types.push_back(seq_type);
        sequence_names.push_back(seq_name);
        sequence_parallel.push_back(seq_parallel);

        index++;
    end
endfunction
```

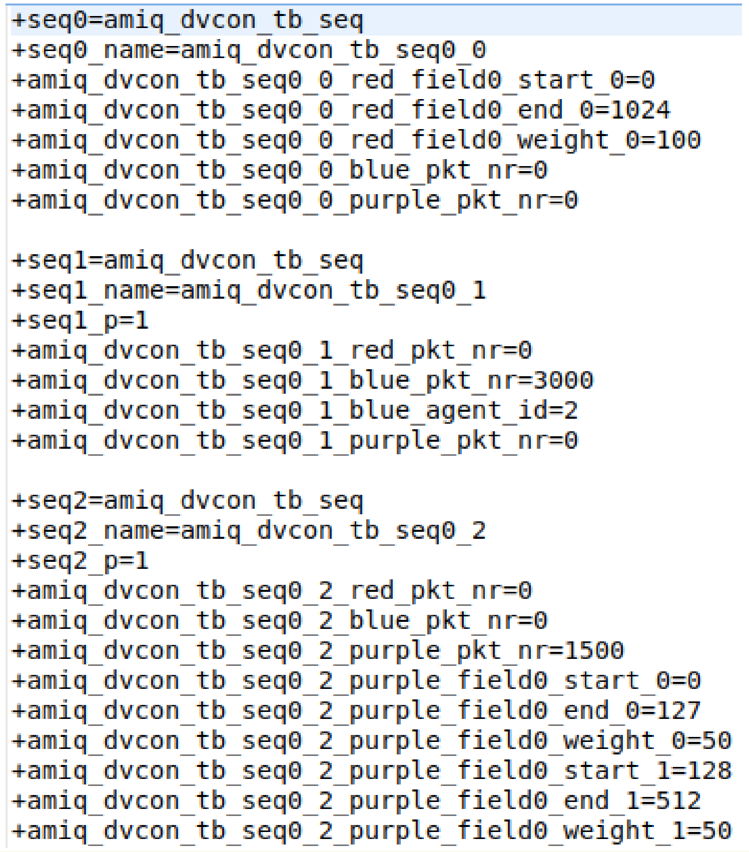


```
task create_and_start_seq(string type_name, string inst_name, int index);
    uvm_object m_object;
    amiq_dvcon_sequence m_sequence;

    // Create
    m_object = factory.create_object_by_name(type_name, this.get_full_name(), inst_name);
    $cast(m_sequence, m_object);

    // If seq for this index is defined but the type is not defined, we throw a warning and continue to next index
    if(m_sequence == null) begin
        `uvm_fatal(get_name(), $sformatf("seq%0d is defined as a type that doesn't exist. Type defined: %0s", index, type_name))
    end

    // Register_all_vars
    m_sequence.register_all_vars();

    // Set the pointer to env and start sequence
    m_sequence.start(virtual_sequencer);

endtask
```

Text inputs vs tests (1)

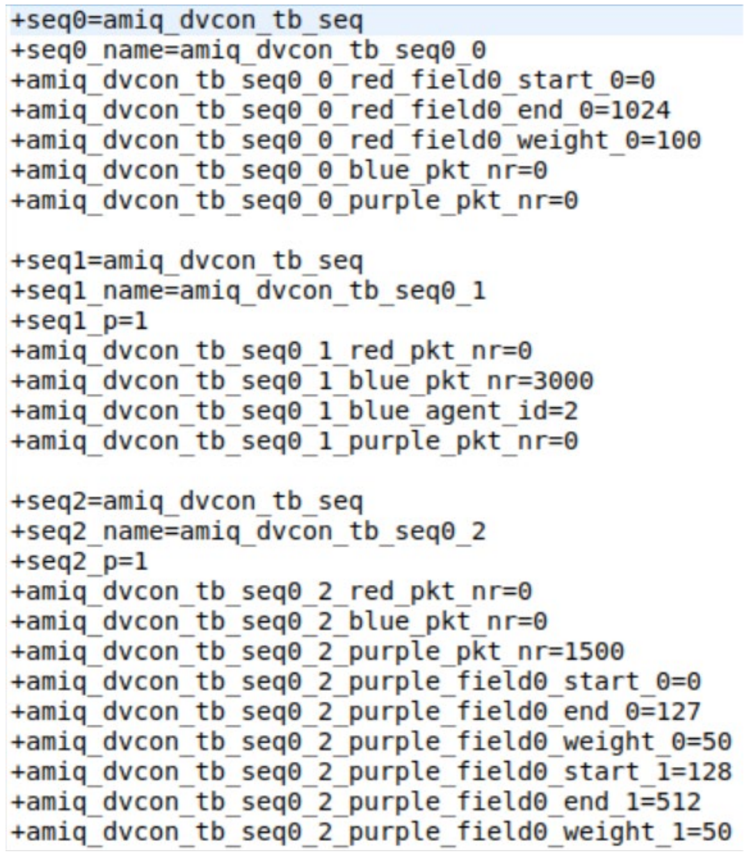


```
class amiq_dvcon_test extends uvm_test;

    // Sequence types
    string sequence_types[$];

    // Sequence names
    string sequence_names[$];

    // Sequence serial/parallel
    bit sequence_parallel[$];

    // Virtual sequencer
    uvm_sequencer virtual_sequencer;

    // Used to create components
    // Needs to be retrieved so it is instantiated globally to reduce performance hit
    uvm_factory factory;

    `uvm_component_utils(amiq_dvcon_test)

    function new(string name = "amiq_dvcon_test", uvm_component parent=null);
        super.new(name,parent);
    endfunction : new

    virtual function void build_phase(uvm_phase phase);
        super.build_phase(phase);
    endfunction : build_phase

    task run_phase(uvm_phase phase);
        super.run_phase(phase);

        if(virtual_sequencer==null) `uvm_fatal(get_name(), $sformatf("A virtual sequence has to be set in the test on which all defined virtual sequences will be started."))

        // Retrieve the factory globally so it is available in all functions without having to "get" it multiple times
        factory = uvm_factory::get();

        // Read the plusargs for all the defined sequences and save their type together with the name and parallelism status, if defined
        retrieve_all_seq_type();

        // Based on the previous returned types, names and parallelism status create and start all sequences
        for(int i=0; i<sequence_types.size(); i++) begin
            schedule_sequence(i);
            wait_threads(i);
        end
    endtask : run_phase
```

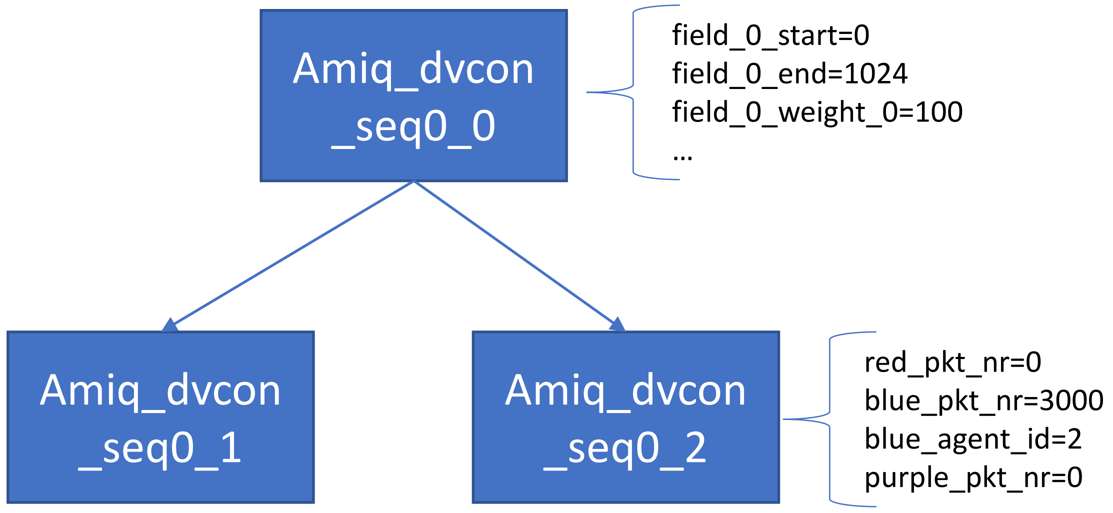


```
+seq0=amiq_dvcon_tb_seq
+seq0_name=amiq_dvcon_tb_seq0_0
+amiq_dvcon_tb_seq0_0_red_field0_start_0=0
+amiq_dvcon_tb_seq0_0_red_field0_end_0=1024
+amiq_dvcon_tb_seq0_0_red_field0_weight_0=100
+amiq_dvcon_tb_seq0_0_blue_pkt_nr=0
+amiq_dvcon_tb_seq0_0_purple_pkt_nr=0

+seq1=amiq_dvcon_tb_seq
+seq1_name=amiq_dvcon_tb_seq0_1
+seq1_p=1
+amiq_dvcon_tb_seq0_1_red_pkt_nr=0
+amiq_dvcon_tb_seq0_1_blue_pkt_nr=3000
+amiq_dvcon_tb_seq0_1_blue_agent_id=2
+amiq_dvcon_tb_seq0_1_purple_pkt_nr=0

+seq2=amiq_dvcon_tb_seq
+seq2_name=amiq_dvcon_tb_seq0_2
+seq2_p=1
+amiq_dvcon_tb_seq0_2_red_pkt_nr=0
+amiq_dvcon_tb_seq0_2_blue_pkt_nr=0
+amiq_dvcon_tb_seq0_2_purple_pkt_nr=1500
+amiq_dvcon_tb_seq0_2_purple_field0_start_0=0
+amiq_dvcon_tb_seq0_2_purple_field0_end_0=127
+amiq_dvcon_tb_seq0_2_purple_field0_weight_0=50
+amiq_dvcon_tb_seq0_2_purple_field0_start_1=128
+amiq_dvcon_tb_seq0_2_purple_field0_end_1=512
+amiq_dvcon_tb_seq0_2_purple_field0_weight_1=50
```

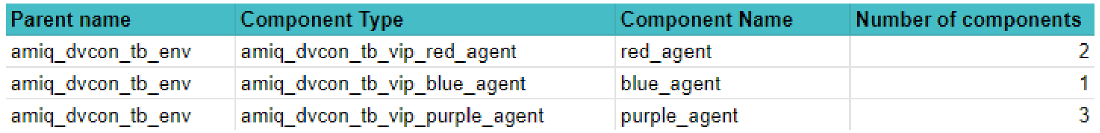
Text inputs vs tests (2)



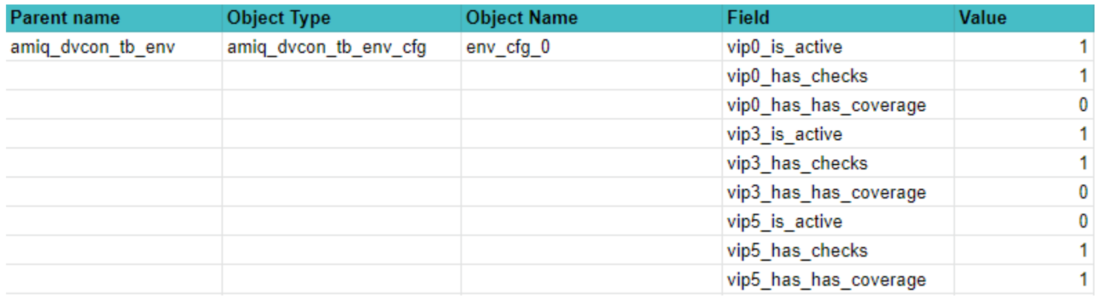
```
+seq0=amiq_dvcon_tb_seq
+seq0_name=amiq_dvcon_tb_seq0_0
+amiq_dvcon_tb_seq0_0_red_field0_start_0=0
+amiq_dvcon_tb_seq0_0_red_field0_end_0=1024
+amiq_dvcon_tb_seq0_0_red_field0_weight_0=100
+amiq_dvcon_tb_seq0_0_blue_pkt_nr=0
+amiq_dvcon_tb_seq0_0_purple_pkt_nr=0

+seq1=amiq_dvcon_tb_seq
+seq1_name=amiq_dvcon_tb_seq0_1
+seq1_p=1
+amiq_dvcon_tb_seq0_1_red_pkt_nr=0
+amiq_dvcon_tb_seq0_1_blue_pkt_nr=3000
+amiq_dvcon_tb_seq0_1_blue_agent_id=2
+amiq_dvcon_tb_seq0_1_purple_pkt_nr=0

+seq2=amiq_dvcon_tb_seq
+seq2_name=amiq_dvcon_tb_seq0_2
+seq2_p=1
+amiq_dvcon_tb_seq0_2_red_pkt_nr=0
+amiq_dvcon_tb_seq0_2_blue_pkt_nr=0
+amiq_dvcon_tb_seq0_2_purple_pkt_nr=1500
+amiq_dvcon_tb_seq0_2_purple_field0_start_0=0
+amiq_dvcon_tb_seq0_2_purple_field0_end_0=127
+amiq_dvcon_tb_seq0_2_purple_field0_weight_0=50
+amiq_dvcon_tb_seq0_2_purple_field0_start_1=128
+amiq_dvcon_tb_seq0_2_purple_field0_end_1=512
+amiq_dvcon_tb_seq0_2_purple_field0_weight_1=50
```



Automation (1)

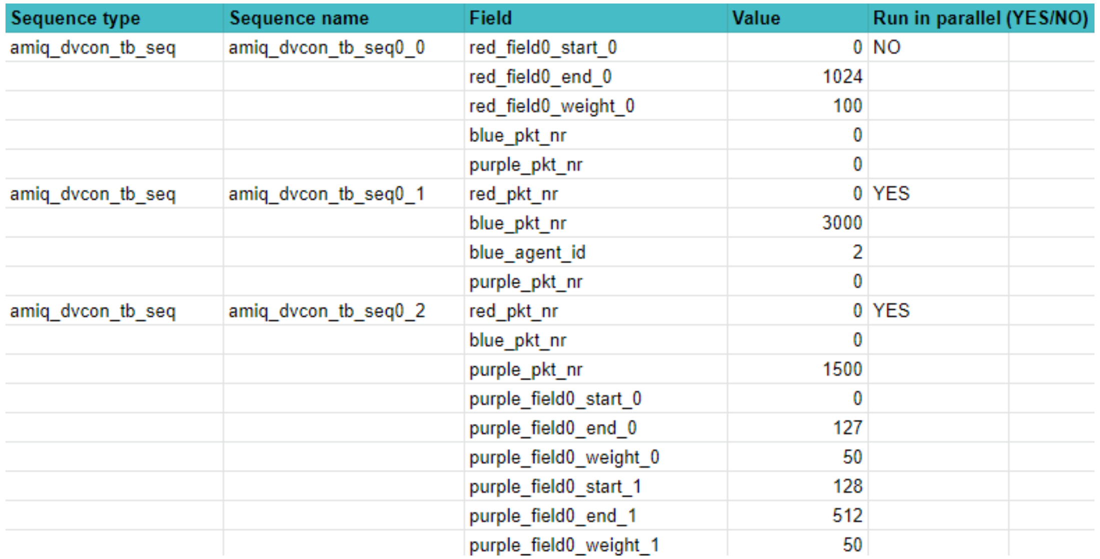


Parent name	Component Type	Component Name	Number of components
amiq_dvcon_tb_env	amiq_dvcon_tb_vip_red_agent	red_agent	2
amiq_dvcon_tb_env	amiq_dvcon_tb_vip_blue_agent	blue_agent	1
amiq_dvcon_tb_env	amiq_dvcon_tb_vip_purple_agent	purple_agent	3

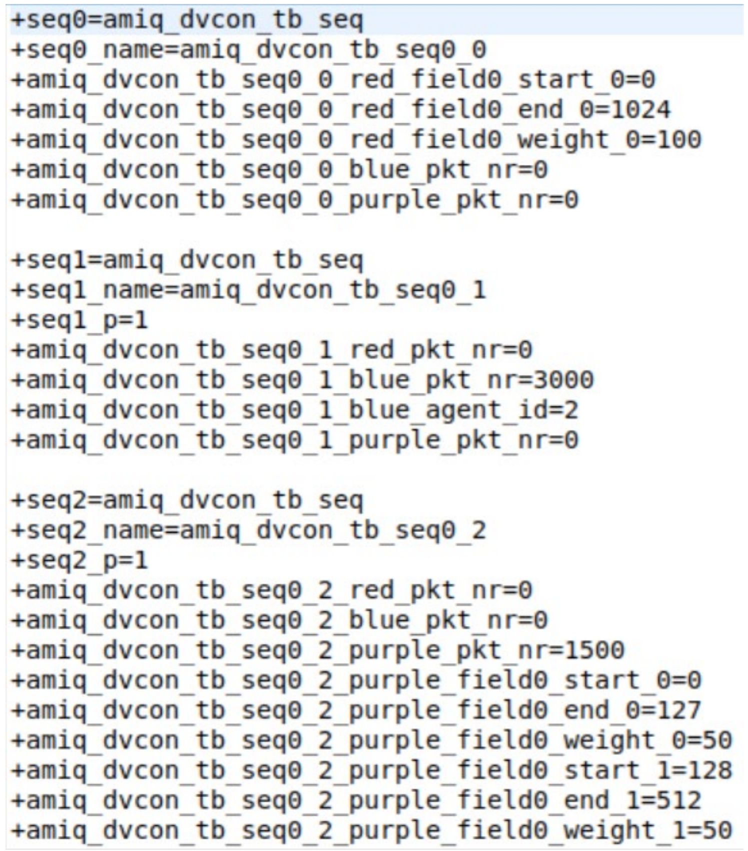


Parent name	Object Type	Object Name	Field	Value
amiq_dvcon_tb_env	amiq_dvcon_tb_env_cfg	env_cfg_0	vip0_is_active	1
			vip0_has_checks	1
			vip0_has_has_coverage	0
			vip3_is_active	1
			vip3_has_checks	1
			vip3_has_has_coverage	0
			vip5_is_active	0
			vip5_has_checks	1
			vip5_has_has_coverage	1

Automation (2)



Sequence type	Sequence name	Field	Value	Run in parallel (YES/NO)
amiq_dvcon_tb_seq	amiq_dvcon_tb_seq0_0	red_field0_start_0	0	NO
		red_field0_end_0	1024	
		red_field0_weight_0	100	
		blue_pkt_nr	0	
		purple_pkt_nr	0	
amiq_dvcon_tb_seq	amiq_dvcon_tb_seq0_1	red_pkt_nr	0	YES
		blue_pkt_nr	3000	
		blue_agent_id	2	
		purple_pkt_nr	0	
amiq_dvcon_tb_seq	amiq_dvcon_tb_seq0_2	red_pkt_nr	0	YES
		blue_pkt_nr	0	
		purple_pkt_nr	1500	
		purple_field0_start_0	0	
		purple_field0_end_0	127	
		purple_field0_weight_0	50	
		purple_field0_start_1	128	
		purple_field0_end_1	512	
		purple_field0_weight_1	50	



```
+seq0=amiq_dvcon_tb_seq
+seq0_name=amiq_dvcon_tb_seq0_0
+amiq_dvcon_tb_seq0_0_red_field0_start_0=0
+amiq_dvcon_tb_seq0_0_red_field0_end_0=1024
+amiq_dvcon_tb_seq0_0_red_field0_weight_0=100
+amiq_dvcon_tb_seq0_0_blue_pkt_nr=0
+amiq_dvcon_tb_seq0_0_purple_pkt_nr=0

+seq1=amiq_dvcon_tb_seq
+seq1_name=amiq_dvcon_tb_seq0_1
+seq1_p=1
+amiq_dvcon_tb_seq0_1_red_pkt_nr=0
+amiq_dvcon_tb_seq0_1_blue_pkt_nr=3000
+amiq_dvcon_tb_seq0_1_blue_agent_id=2
+amiq_dvcon_tb_seq0_1_purple_pkt_nr=0

+seq2=amiq_dvcon_tb_seq
+seq2_name=amiq_dvcon_tb_seq0_2
+seq2_p=1
+amiq_dvcon_tb_seq0_2_red_pkt_nr=0
+amiq_dvcon_tb_seq0_2_blue_pkt_nr=0
+amiq_dvcon_tb_seq0_2_purple_pkt_nr=1500
+amiq_dvcon_tb_seq0_2_purple_field0_start_0=0
+amiq_dvcon_tb_seq0_2_purple_field0_end_0=127
+amiq_dvcon_tb_seq0_2_purple_field0_weight_0=50
+amiq_dvcon_tb_seq0_2_purple_field0_start_1=128
+amiq_dvcon_tb_seq0_2_purple_field0_end_1=512
+amiq_dvcon_tb_seq0_2_purple_field0_weight_1=50
```

Conclusions

- Simplicity is key to scalability
- Drawing outside the lines can be an inconvenient in engineering
- Verification is a puzzle, not a painting
- Past a certain threshold of scale, maintenance effort supersedes debug

What are we working on

- Further automation on TB creation
- Debug helper
- Ranking based on runtime events
- Feedback loop (DVCon US 2023)

Questions

SYSTEMS INITIATIVE