Ripple Carry Adder Easier UVM

Author: Vladimir Armstrong
vladimirarmstrong@opencores.org

Rev. 1.1
April 14, 2019

This page has been intentionally left blank.

Revision History

Rev.	Date	Author	Description
1.0	03/19/19	Vladimir Armstrong	First Draft
1.1	4/13/19	Vladimir Armstrong	Added UVM testbench source files

Contents

INTRODUCTION **1**

UVM ARCHITECTURE SPECIFICATIONS **2**

UVM CODE GENERATOR **3**

TEMPLET FILES **4**

INCLUDE FILE: *RCA.TPL* 4
DRIVER FILE: *RCA_DRIVER_INC.SV* 5
MONITOR FILE: *RCA_MONITOR_INC.SV* 5
PIN LIST FILE: *PINLIST* 5
COMMON TEMPLET FILE: *COMMON.TPL* 5
DUT FILE: *DESIGN.SV* 6

UVM TESTBENCH **7**

TOP_TB **9**

TOP_TB.SV 9
TOP_TH.SV 11

TOP_TEST **12**

TOP_TEST_PKG.SV 12
TOP_TEST.SV 13

TOP **14**

TOP_CONFIG.SV 14
TOP_ENV.SV 16
TOP_PKG.SV 19
TOP_SEQ_LIB.SV 20

RCA **22**

RCA_AGENT.SV 22
RCA_CONFIG.SV 25
RCA_COVERAGE.SV 26
RCA_DRIVER.SV 28
RCA_IF.SV 29
RCA_MONITOR.SV 30
RCA_PKG.SV 32
RCA_SEQ_LIB.SV 33
RCA_SEQUENCER.SV 34
RCA_TRANS.SV 35

INCLUDE **38**

RCA_DRIVER_INC.SV 38
RCA_MONITOR_INC.SV 39

APPENDIX A **40**

INDEX.. 41

1 Introduction

This document describes the verification of Ripple Carry Adder RTL module [2] by using a minimal of Easier UVM Code Generator [1] to keep it simple. The verification flow has 4 basic steps and is shown in Figure 1; starting with UVM architecture specifications Figure 2 from which a templet files [4] are created which are used as input to Perl script Figure 3 which outputs System Verilog UVM testbench Figure 4.

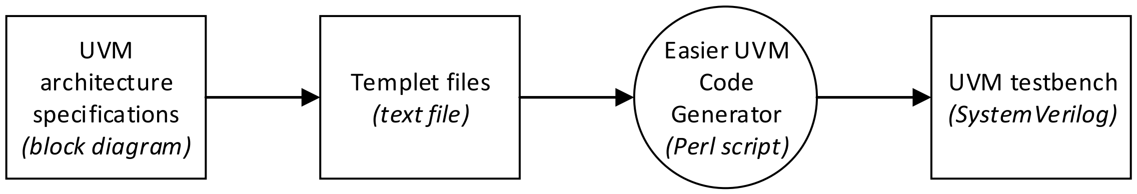
UVM architecture specifications (block diagram)
Templet files (text file)
Easier UVM Code Generator (Perl script)
UVM testbench (SystemVerilog)

Figure 1 Easier UVM verification flow.

2

UVM Architecture Specifications

The UVM architecture is specified in Figure 2, to keep it simple Scoreboard or Reference Model is not used.

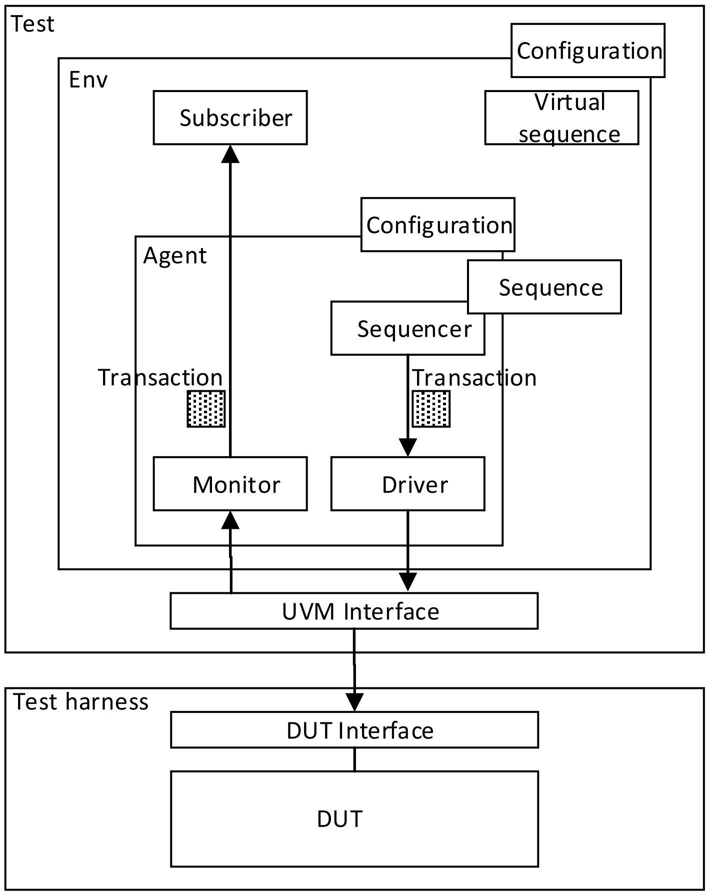

Figure 2 UVM architecture specifications.

3

UVM Code Generator

The Easier UVM Code Generator Perl script inputs 6 templet files [4] and outputs UVM testbench is shown in Figure 3.

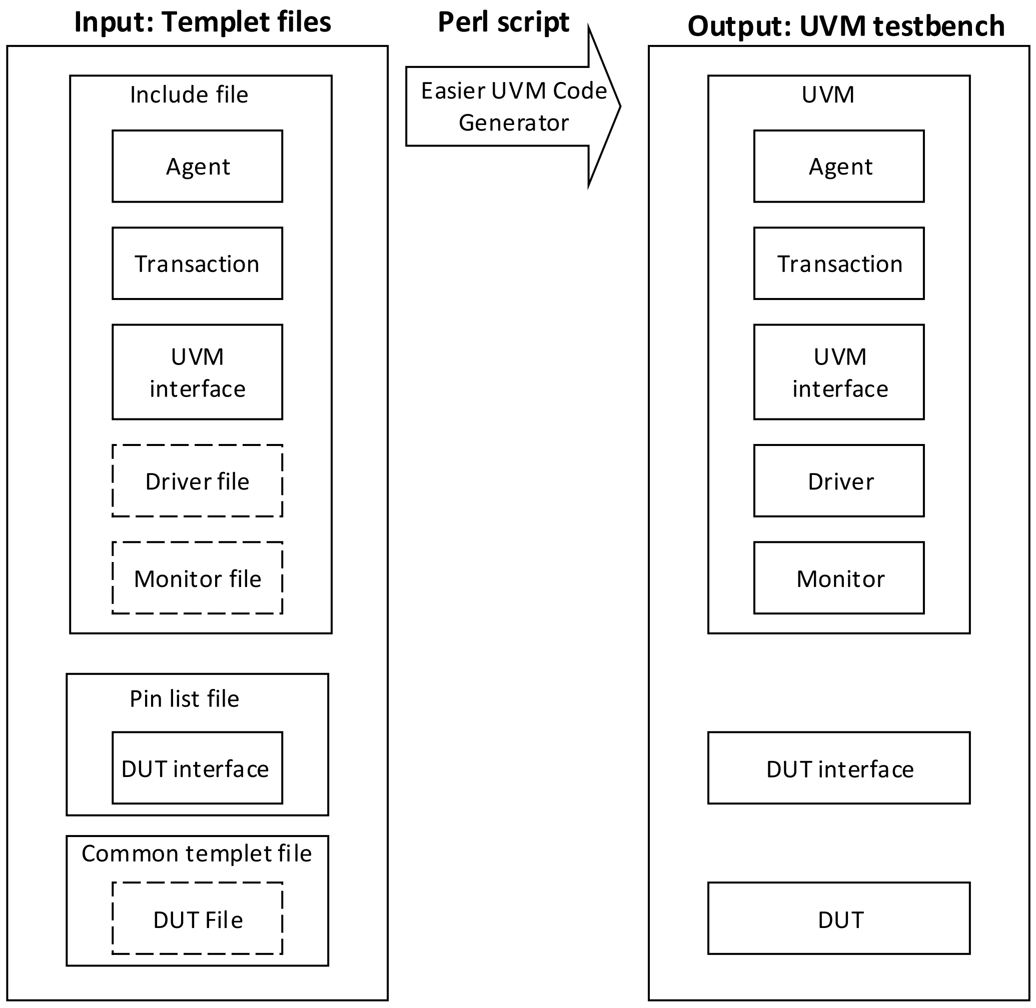

Figure 3 Easier UVM Code Generator.

4

Templet Files

Include File: *rca.tpl*

```
# Agent
agent_name = rca

# Transaction
trans_item = trans
trans_var = rand logic [15:0] input1;
trans_var = rand logic [15:0] input2;
trans_var = rand logic carryinput;
trans_var = logic carryoutput;
trans_var = logic [15:0] sum;

# Constraint
trans_var  = constraint c_addr_a { 0 <= input1; input1 < 5; }
trans_var  = constraint c_addr_b { 0 <= input2; input2 < 5; }

# UVM Interface
if_port = logic [15:0] a;
if_port = logic [15:0] b;
if_port = logic ci;
if_port = logic co;
if_port = logic [15:0] s;
if_port = logic clk;

# Test Clock
if_clock = clk

# Driver and Monitor pointer
driver_inc = rca_driver_inc.sv inline
monitor_inc = rca_monitor_inc.sv inline
```

Driver File: *rca_driver_inc.sv*

```
task rca_driver::do_drive();
  vif.a <= req.input1;
  vif.b <= req.input2;
  vif.ci <= req.carryinput;
  @(posedge vif.clk);
endtask
```

Monitor File: *rca_monitor_inc.sv*

```
task rca_monitor::do_mon;
  forever @(posedge vif.clk)
    begin
      m_trans.input1 = vif.a;
      m_trans.input2 = vif.b;
      m_trans.carryinput = vif.ci;
      m_trans.carryoutput = vif.co;
      m_trans.sum = vif.s;
      analysis_port.write(m_trans);
      `uvm_info(get_type_name(),$sformatf("a(%0d) + b(%0d) + ci(%0d) =
co(%0d) and s(%0d)", vif.a, vif.b, vif.ci, vif.co, vif.s), UVM_MEDIUM);
    end
endtask
```

Pin List File: *pinlist*

```
!rca_if
a a
b b
ci ci
co co
s s
```

Common Templet File: *common.tpl*

```
dut_top = rca
top_default_seq_count = 8
```

DUT File: *design.sv*

```
module rca(
           input [15:0]       a,
           input [15:0]       b,
           input              ci, // Carry Input

           output logic       co, // Carry Output
           output logic [15:0] s // Sum
           );

   logic                        a0,a1,a2,a3,a4,a5,a6,a7;
   logic                        a8,a9,a10,a11,a12,a13,a14,a15;
   logic                        b0,b1,b2,b3,b4,b5,b6,b7;
   logic                        b8,b9,b10,b11,b12,b13,b14,b15;
   logic                        c0,c1,c2,c3,c4,c5,c6,c7;
   logic                        c8,c9,c10,c11,c12,c13,c14;
   logic                        s0,s1,s2,s3,s4,s5,s6,s7;
   logic                        s8,s9,s10,s11,s12,s13,s14,s15;

   assign a0 = a[0], a1 = a[1], a2 = a[2], a3 = a[3];
   assign a4 = a[4], a5 = a[5], a6 = a[6], a7 = a[7];
   assign a8 = a[8], a9 = a[9], a10 = a[10];
   assign a11 = a[11], a12 = a[12], a13 = a[13];
   assign a14 = a[14], a15 = a[15];
   assign b0 = b[0], b1 = b[1], b2 = b[2], b3 = b[3];
   assign b4 = b[4], b5 = b[5], b6 = b[6], b7 = b[7];
   assign b8 = b[8], b9 = b[9], b10 = b[10], b11 = b[11];
   assign b12 = b[12], b13 = b[13], b14 = b[14], b15 = b[15];
   assign s[0] = s0, s[1] = s1, s[2] = s2, s[3] = s3;
   assign s[4] = s4, s[5] = s5,s[6] = s6, s[7] = s7;
   assign s[8] = s8, s[9] = s9, s[10] = s10, s[11] = s11;
   assign s[12] = s12, s[13] = s13, s[14] = s14, s[15] = s15;

   fa fa_inst0(.a(a0),.b(b0),.ci(ci),.co(c0),.s(s0));
   fa fa_inst1(.a(a1),.b(b1),.ci(c0),.co(c1),.s(s1));
   fa fa_inst2(.a(a2),.b(b2),.ci(c1),.co(c2),.s(s2));
   fa fa_inst3(.a(a3),.b(b3),.ci(c2),.co(c3),.s(s3));
   fa fa_inst4(.a(a4),.b(b4),.ci(c3),.co(c4),.s(s4));
   fa fa_inst5(.a(a5),.b(b5),.ci(c4),.co(c5),.s(s5));
   fa fa_inst6(.a(a6),.b(b6),.ci(c5),.co(c6),.s(s6));
   fa fa_inst7(.a(a7),.b(b7),.ci(c6),.co(c7),.s(s7));
   fa fa_inst8(.a(a8),.b(b8),.ci(c7),.co(c8),.s(s8));
   fa fa_inst9(.a(a9),.b(b9),.ci(c8),.co(c9),.s(s9));
   fa fa_inst10(.a(a10),.b(b10),.ci(c9),.co(c10),.s(s10));
   fa fa_inst11(.a(a11),.b(b11),.ci(c10),.co(c11),.s(s11));
   fa fa_inst12(.a(a12),.b(b12),.ci(c11),.co(c12),.s(s12));
   fa fa_inst13(.a(a13),.b(b13),.ci(c12),.co(c13),.s(s13));
   fa fa_inst14(.a(a14),.b(b14),.ci(c13),.co(c14),.s(s14));
   fa fa_inst15(.a(a15),.b(b15),.ci(c14),.co(co),.s(s15));

endmodule
```

5

UVM Testbench

```
--------------------------------------------------------------
Name                        Type                      Size  Value
--------------------------------------------------------------
uvm_test_top                top_test                  -     @344
  m_env                     top_env                   -     @357
    m_rca_agent             rca_agent                 -     @373
      analysis_port         uvm_analysis_port         -     @382
      m_driver              rca_driver                -     @432
        rsp_port            uvm_analysis_port         -     @451
        seq_item_port       uvm_seq_item_pull_port    -     @441
      m_monitor             rca_monitor               -     @412
        analysis_port       uvm_analysis_port         -     @421
      m_sequencer           uvm_sequencer             -     @461
        rsp_export          uvm_analysis_export       -     @470
        seq_item_export     uvm_seq_item_pull_imp     -     @588
        arbitration_queue   array                     0     -
        lock_queue          array                     0     -
        num_last_reqs       integral                  32    'd1
        num_last_rsps       integral                  32    'd1
    m_rca_coverage          rca_coverage              -     @392
      analysis_imp          uvm_analysis_imp          -     @401
--------------------------------------------------------------
```

Figure 4 UVM testbench summary

```
tb
├── include
│   ├── rca_driver_inc.sv
│   └── rca_monitor_inc.sv
├── rca
│   └── sv
│       ├── rca_agent.sv
│       ├── rca_config.sv
│       ├── rca_coverage.sv
│       ├── rca_driver.sv
│       ├── rca_if.sv
│       ├── rca_monitor.sv
│       ├── rca_pkg.sv
│       ├── rca_seq_lib.sv
│       ├── rca_sequencer.sv
│       └── rca_trans.sv
├── top
│   └── sv
│       ├── top_config.sv
│       ├── top_env.sv
│       ├── top_pkg.sv
│       └── top_seq_lib.sv
├── top_tb
│   └── sv
│       ├── top_tb.sv
│       └── top_th.sv
└── top_test
    └── sv
        ├── top_test_pkg.sv
        └── top_test.sv
```

Figure 5 UVM testbench directory structure

6

top_tb

top_tb.sv

```
module top_tb;

  timeunit      1ns;
  timeprecision 1ps;

  `include "uvm_macros.svh"

  import uvm_pkg::*;

  import top_test_pkg::*;
  import top_pkg::top_config;

  // Configuration object for top-level environment
  top_config top_env_config;

  // Test harness
  top_th th();

  // You can insert code here by setting tb_inc_inside_module in file
common.tpl

  // You can remove the initial block below by setting
tb_generate_run_test = no in file common.tpl

  initial
  begin
    // You can insert code here by setting tb_prepend_to_initial in file
common.tpl

    // Create and populate top-level configuration object
    top_env_config = new("top_env_config");
    if ( !top_env_config.randomize() )
      `uvm_error("top_tb", "Failed to randomize top-level configuration
object" )

    top_env_config.rca_vif                = th.rca_if_0;
    top_env_config.is_active_rca          = UVM_ACTIVE;
    top_env_config.checks_enable_rca   = 1;
    top_env_config.coverage_enable_rca = 1;


    uvm_config_db #(top_config)::set(null, "uvm_test_top", "config",
top_env_config);
    uvm_config_db #(top_config)::set(null, "uvm_test_top.m_env",
"config", top_env_config);

    // You can insert code here by setting tb_inc_before_run_test in
file common.tpl

    run_test();
  end

endmodule
```

top_th.sv

```
module top_th;

  timeunit      1ns;
  timeprecision 1ps;


  // You can remove clock and reset below by setting
th_generate_clock_and_reset = no in file common.tpl

  // Example clock and reset declarations
  logic clock = 0;
  logic reset;

  // Example clock generator process
  always #10 clock = ~clock;

  // Example reset generator process
  initial
  begin
    reset = 0;         // Active low reset in this example
    #75 reset = 1;
  end

  assign rca_if_0.clk = clock;

  // You can insert code here by setting th_inc_inside_module in file
common.tpl

  // Pin-level interfaces connected to DUT
  // You can remove interface instances by setting
generate_interface_instance = no in the interface template file

  rca_if  rca_if_0 ();

  rca uut (
    .a (rca_if_0.a),
    .b (rca_if_0.b),
    .ci(rca_if_0.ci),
    .co(rca_if_0.co),
    .s (rca_if_0.s)
  );

endmodule
```

7

top_test

top_test_pkg.sv

```
package top_test_pkg;

  `include "uvm_macros.svh"

  import uvm_pkg::*;

  import rca_pkg::*;
  import top_pkg::*;

  `include "top_test.sv"

endpackage : top_test_pkg
```

top_test.sv

```
// You can insert code here by setting test_inc_before_class in file
common.tpl

class top_test extends uvm_test;

  `uvm_component_utils(top_test)

  top_env m_env;

  extern function new(string name, uvm_component parent);

  // You can remove build_phase method by setting
test_generate_methods_inside_class = no in file common.tpl

  extern function void build_phase(uvm_phase phase);

  // You can insert code here by setting test_inc_inside_class in file
common.tpl

endclass : top_test


function top_test::new(string name, uvm_component parent);
  super.new(name, parent);
endfunction : new


// You can remove build_phase method by setting
test_generate_methods_after_class = no in file common.tpl

function void top_test::build_phase(uvm_phase phase);

  // You can insert code here by setting test_prepend_to_build_phase in
file common.tpl

  // You could modify any test-specific configuration object variables
here



  m_env = top_env::type_id::create("m_env", this);

  // You can insert code here by setting test_append_to_build_phase in
file common.tpl

endfunction : build_phase


// You can insert code here by setting test_inc_after_class in file
common.tpl
```

8 top

top_config.sv

```
// You can insert code here by setting top_env_config_inc_before_class
in file common.tpl

class top_config extends uvm_object;

  // Do not register config class with the factory

  virtual rca_if           rca_vif;

  uvm_active_passive_enum  is_active_rca        = UVM_ACTIVE;
  bit                      checks_enable_rca;
  bit                      coverage_enable_rca;

  // You can insert variables here by setting config_var in file
common.tpl

  // You can remove new by setting
top_env_config_generate_methods_inside_class = no in file common.tpl

  extern function new(string name = "");

  // You can insert code here by setting top_env_config_inc_inside_class
in file common.tpl

endclass : top_config


// You can remove new by setting
top_env_config_generate_methods_after_class = no in file common.tpl

function top_config::new(string name = "");
  super.new(name);

  // You can insert code here by setting top_env_config_append_to_new in
file common.tpl

endfunction : new
```

```
// You can insert code here by setting top_env_config_inc_after_class in
file common.tpl
```

top_env.sv

```
// You can insert code here by setting top_env_inc_before_class in file
common.tpl

class top_env extends uvm_env;

  `uvm_component_utils(top_env)

  extern function new(string name, uvm_component parent);


  // Child agents
  rca_config    m_rca_config;
  rca_agent     m_rca_agent;
  rca_coverage  m_rca_coverage;

  top_config    m_config;

  // You can remove build/connect/run_phase by setting
top_env_generate_methods_inside_class = no in file common.tpl

  extern function void build_phase(uvm_phase phase);
  extern function void connect_phase(uvm_phase phase);
  extern function void end_of_elaboration_phase(uvm_phase phase);
  extern task          run_phase(uvm_phase phase);

  // You can insert code here by setting top_env_inc_inside_class in
file common.tpl

endclass : top_env


function top_env::new(string name, uvm_component parent);
  super.new(name, parent);
endfunction : new


// You can remove build/connect/run_phase by setting
top_env_generate_methods_after_class = no in file common.tpl

function void top_env::build_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "In build_phase", UVM_HIGH)

  // You can insert code here by setting top_env_prepend_to_build_phase
in file common.tpl

  if (!uvm_config_db #(top_config)::get(this, "", "config", m_config))
    `uvm_error(get_type_name(), "Unable to get top_config")

  m_rca_config                 = new("m_rca_config");
  m_rca_config.vif             = m_config.rca_vif;
  m_rca_config.is_active       = m_config.is_active_rca;
  m_rca_config.checks_enable   = m_config.checks_enable_rca;
  m_rca_config.coverage_enable = m_config.coverage_enable_rca;
  // You can insert code here by setting agent_copy_config_vars in file
rca.tpl

  uvm_config_db #(rca_config)::set(this, "m_rca_agent", "config",
m_rca_config);
  if (m_rca_config.is_active == UVM_ACTIVE )
    uvm_config_db #(rca_config)::set(this, "m_rca_agent.m_sequencer",
"config", m_rca_config);
  uvm_config_db #(rca_config)::set(this, "m_rca_coverage", "config",
m_rca_config);


  m_rca_agent    = rca_agent   ::type_id::create("m_rca_agent", this);
  m_rca_coverage = rca_coverage::type_id::create("m_rca_coverage",
this);

  // You can insert code here by setting top_env_append_to_build_phase
in file common.tpl

endfunction : build_phase


function void top_env::connect_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "In connect_phase", UVM_HIGH)

  m_rca_agent.analysis_port.connect(m_rca_coverage.analysis_export);


  // You can insert code here by setting top_env_append_to_connect_phase
in file common.tpl

endfunction : connect_phase


// You can remove end_of_elaboration_phase by setting
top_env_generate_end_of_elaboration = no in file common.tpl

function void top_env::end_of_elaboration_phase(uvm_phase phase);
  uvm_factory factory = uvm_factory::get();
  `uvm_info(get_type_name(), "Information printed from
top_env::end_of_elaboration_phase method", UVM_MEDIUM)
  `uvm_info(get_type_name(), $sformatf("Verbosity threshold is %d",
get_report_verbosity_level()), UVM_MEDIUM)
  uvm_top.print_topology();
  factory.print();
endfunction : end_of_elaboration_phase


// You can remove run_phase by setting top_env_generate_run_phase = no
in file common.tpl

task top_env::run_phase(uvm_phase phase);
  top_default_seq vseq;
  vseq = top_default_seq::type_id::create("vseq");
  vseq.set_item_context(null, null);
  if ( !vseq.randomize() )
```

```
    `uvm_fatal(get_type_name(), "Failed to randomize virtual sequence")
  vseq.m_rca_agent = m_rca_agent;
  vseq.set_starting_phase(phase);
  vseq.start(null);

  // You can insert code here by setting top_env_append_to_run_phase in
file common.tpl

endtask : run_phase


// You can insert code here by setting top_env_inc_after_class in file
common.tpl
```

top_pkg.sv

```
package top_pkg;

  `include "uvm_macros.svh"

  import uvm_pkg::*;

  import rca_pkg::*;

  `include "top_config.sv"
  `include "top_seq_lib.sv"
  `include "top_env.sv"

endpackage : top_pkg
```

top_seq_lib.sv

```
class top_default_seq extends uvm_sequence #(uvm_sequence_item);

  `uvm_object_utils(top_default_seq)

  rca_agent  m_rca_agent;

  // Number of times to repeat child sequences
  int m_seq_count = 8;

  extern function new(string name = "");
  extern task body();
  extern task pre_start();
  extern task post_start();

`ifndef UVM_POST_VERSION_1_1
  // Functions to support UVM 1.2 objection API in UVM 1.1
  extern function uvm_phase get_starting_phase();
  extern function void set_starting_phase(uvm_phase phase);
`endif

endclass : top_default_seq


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


task top_default_seq::body();
  `uvm_info(get_type_name(), "Default sequence starting", UVM_HIGH)


  repeat (m_seq_count)
  begin
    fork
      if (m_rca_agent.m_config.is_active == UVM_ACTIVE)
      begin
        rca_default_seq seq;
        seq = rca_default_seq::type_id::create("seq");
        seq.set_item_context(this, m_rca_agent.m_sequencer);
        if ( !seq.randomize() )
          `uvm_error(get_type_name(), "Failed to randomize sequence")
        seq.set_starting_phase( get_starting_phase() );
        seq.start(m_rca_agent.m_sequencer, this);
      end
    join
  end

  `uvm_info(get_type_name(), "Default sequence completed", UVM_HIGH)
endtask : body


task top_default_seq::pre_start();
```

```
  uvm_phase phase = get_starting_phase();
  if (phase != null)
    phase.raise_objection(this);
endtask: pre_start


task top_default_seq::post_start();
  uvm_phase phase = get_starting_phase();
  if (phase != null)
    phase.drop_objection(this);
endtask: post_start


`ifndef UVM_POST_VERSION_1_1
function uvm_phase top_default_seq::get_starting_phase();
  return starting_phase;
endfunction: get_starting_phase


function void top_default_seq::set_starting_phase(uvm_phase phase);
  starting_phase = phase;
endfunction: set_starting_phase
`endif


// You can insert code here by setting top_seq_inc in file common.tpl
```

9

rca

rca_agent.sv

```
// You can insert code here by setting agent_inc_before_class in file
rca.tpl

class rca_agent extends uvm_agent;

  `uvm_component_utils(rca_agent)

  uvm_analysis_port #(trans) analysis_port;

  rca_config        m_config;
  rca_sequencer_t   m_sequencer;
  rca_driver        m_driver;
  rca_monitor       m_monitor;

  local int m_is_active = -1;

  extern function new(string name, uvm_component parent);

  // You can remove build/connect_phase and get_is_active by setting
agent_generate_methods_inside_class = no in file rca.tpl

  extern function void build_phase(uvm_phase phase);
  extern function void connect_phase(uvm_phase phase);
  extern function uvm_active_passive_enum get_is_active();

  // You can insert code here by setting agent_inc_inside_class in file
rca.tpl

endclass : rca_agent


function  rca_agent::new(string name, uvm_component parent);
  super.new(name, parent);
  analysis_port = new("analysis_port", this);
endfunction : new
```

```
// You can remove build/connect_phase and get_is_active by setting
agent_generate_methods_after_class = no in file rca.tpl

function void rca_agent::build_phase(uvm_phase phase);

  // You can insert code here by setting agent_prepend_to_build_phase in
file rca.tpl

  if (!uvm_config_db #(rca_config)::get(this, "", "config", m_config))
    `uvm_error(get_type_name(), "rca config not found")

  m_monitor     = rca_monitor    ::type_id::create("m_monitor", this);

  if (get_is_active() == UVM_ACTIVE)
  begin
    m_driver    = rca_driver     ::type_id::create("m_driver", this);
    m_sequencer = rca_sequencer_t::type_id::create("m_sequencer", this);
  end

  // You can insert code here by setting agent_append_to_build_phase in
file rca.tpl

endfunction : build_phase


function void rca_agent::connect_phase(uvm_phase phase);
  if (m_config.vif == null)
    `uvm_warning(get_type_name(), "rca virtual interface is not set!")

  m_monitor.vif = m_config.vif;
  m_monitor.analysis_port.connect(analysis_port);

  if (get_is_active() == UVM_ACTIVE)
  begin
    m_driver.seq_item_port.connect(m_sequencer.seq_item_export);
    m_driver.vif = m_config.vif;
  end

  // You can insert code here by setting agent_append_to_connect_phase
in file rca.tpl

endfunction : connect_phase


function uvm_active_passive_enum rca_agent::get_is_active();
  if (m_is_active == -1)
  begin
    if (uvm_config_db#(uvm_bitstream_t)::get(this, "", "is_active",
m_is_active))
    begin
      if (m_is_active != m_config.is_active)
        `uvm_warning(get_type_name(), "is_active field in config_db
conflicts with config object")
    end
    else
      m_is_active = m_config.is_active;
  end
  return uvm_active_passive_enum'(m_is_active);
endfunction : get_is_active


// You can insert code here by setting agent_inc_after_class in file
rca.tpl
```

rca_config.sv

```
// You can insert code here by setting agent_config_inc_before_class in
file rca.tpl

class rca_config extends uvm_object;

  // Do not register config class with the factory

  virtual rca_if           vif;

  uvm_active_passive_enum  is_active = UVM_ACTIVE;
  bit                      coverage_enable;
  bit                      checks_enable;

  // You can insert variables here by setting config_var in file rca.tpl

  // You can remove new by setting
agent_config_generate_methods_inside_class = no in file rca.tpl

  extern function new(string name = "");

  // You can insert code here by setting agent_config_inc_inside_class
in file rca.tpl

endclass : rca_config


// You can remove new by setting
agent_config_generate_methods_after_class = no in file rca.tpl

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


// You can insert code here by setting agent_config_inc_after_class in
file rca.tpl
```

rca_coverage.sv

```
// You can insert code here by setting agent_cover_inc_before_class in
file rca.tpl

class rca_coverage extends uvm_subscriber #(trans);

  `uvm_component_utils(rca_coverage)

  rca_config m_config;
  bit        m_is_covered;
  trans      m_item;

  // You can replace covergroup m_cov by setting agent_cover_inc in file
rca.tpl
  // or remove covergroup m_cov by setting
agent_cover_generate_methods_inside_class = no in file rca.tpl

  covergroup m_cov;
    option.per_instance = 1;
    // You may insert additional coverpoints here ...

    cp_input1: coverpoint m_item.input1;
    //  Add bins here if required

    cp_input2: coverpoint m_item.input2;
    //  Add bins here if required

    cp_carryinput: coverpoint m_item.carryinput;
    //  Add bins here if required

    cp_carryoutput: coverpoint m_item.carryoutput;
    //  Add bins here if required

    cp_sum: coverpoint m_item.sum;
    //  Add bins here if required

  endgroup

  // You can remove new, write, and report_phase by setting
agent_cover_generate_methods_inside_class = no in file rca.tpl

  extern function new(string name, uvm_component parent);
  extern function void write(input trans t);
  extern function void build_phase(uvm_phase phase);
  extern function void report_phase(uvm_phase phase);

  // You can insert code here by setting agent_cover_inc_inside_class in
file rca.tpl

endclass : rca_coverage


// You can remove new, write, and report_phase by setting
agent_cover_generate_methods_after_class = no in file rca.tpl
```

```
function rca_coverage::new(string name, uvm_component parent);
  super.new(name, parent);
  m_is_covered = 0;
  m_cov = new();
endfunction : new


function void rca_coverage::write(input trans t);
  m_item = t;
  if (m_config.coverage_enable)
  begin
    m_cov.sample();
    // Check coverage - could use m_cov.option.goal instead of 100 if
your simulator supports it
    if (m_cov.get_inst_coverage() >= 100) m_is_covered = 1;
  end
endfunction : write


function void rca_coverage::build_phase(uvm_phase phase);
  if (!uvm_config_db #(rca_config)::get(this, "", "config", m_config))
    `uvm_error(get_type_name(), "rca config not found")
endfunction : build_phase


function void rca_coverage::report_phase(uvm_phase phase);
  if (m_config.coverage_enable)
    `uvm_info(get_type_name(), $sformatf("Coverage score = %3.1f%%",
m_cov.get_inst_coverage()), UVM_MEDIUM)
  else
    `uvm_info(get_type_name(), "Coverage disabled for this agent",
UVM_MEDIUM)
endfunction : report_phase


// You can insert code here by setting agent_cover_inc_after_class in
file rca.tpl
```

rca_driver.sv

```
// You can insert code here by setting driver_inc_before_class in file
rca.tpl

class rca_driver extends uvm_driver #(trans);

  `uvm_component_utils(rca_driver)

  virtual rca_if vif;

  extern function new(string name, uvm_component parent);

  // Methods run_phase and do_drive generated by setting driver_inc in
file rca.tpl
  extern task run_phase(uvm_phase phase);
  extern task do_drive();

  // You can insert code here by setting driver_inc_inside_class in file
rca.tpl

endclass : rca_driver


function rca_driver::new(string name, uvm_component parent);
  super.new(name, parent);
endfunction : new


task rca_driver::run_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "run_phase", UVM_HIGH)

  forever
  begin
    seq_item_port.get_next_item(req);
      `uvm_info(get_type_name(), {"req item\n",req.sprint}, UVM_HIGH)
    do_drive();
    seq_item_port.item_done();
  end
endtask : run_phase


// Start of inlined include file
generated_tb/tb/include/rca_driver_inc.sv
task rca_driver::do_drive();
  vif.a <= req.input1;
  vif.b <= req.input2;
  vif.ci <= req.carryinput;
  @(posedge vif.clk);
endtask// End of inlined include file

// You can insert code here by setting driver_inc_after_class in file
rca.tpl
```

rca_if.sv

```
interface rca_if();

  timeunit      1ns;
  timeprecision 1ps;

  import rca_pkg::*;

  logic [15:0] a;
  logic [15:0] b;
  logic ci;
  logic co;
  logic [15:0] s;
  logic clk;

  // You can insert properties and assertions here

  // You can insert code here by setting if_inc_inside_interface in file
rca.tpl

endinterface : rca_if
```

rca_monitor.sv

```
// You can insert code here by setting monitor_inc_before_class in file
rca.tpl

class rca_monitor extends uvm_monitor;

  `uvm_component_utils(rca_monitor)

  virtual rca_if vif;

  uvm_analysis_port #(trans) analysis_port;

  trans m_trans;

  extern function new(string name, uvm_component parent);

  // Methods build_phase, run_phase, and do_mon generated by setting
monitor_inc in file rca.tpl
  extern function void build_phase(uvm_phase phase);
  extern task run_phase(uvm_phase phase);
  extern task do_mon();

  // You can insert code here by setting monitor_inc_inside_class in
file rca.tpl

endclass : rca_monitor


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


function void rca_monitor::build_phase(uvm_phase phase);
endfunction : build_phase


task rca_monitor::run_phase(uvm_phase phase);
  `uvm_info(get_type_name(), "run_phase", UVM_HIGH)

  m_trans = trans::type_id::create("m_trans");
  do_mon();
endtask : run_phase


// Start of inlined include file
generated_tb/tb/include/rca_monitor_inc.sv
task rca_monitor::do_mon;
  forever @(posedge vif.clk)
    begin
      m_trans.input1 = vif.a;
      m_trans.input2 = vif.b;
      m_trans.carryinput = vif.ci;
      m_trans.carryoutput = vif.co;
      m_trans.sum = vif.s;
      analysis_port.write(m_trans);
      `uvm_info(get_type_name(),$sformatf("a(%0d) + b(%0d) + ci(%0d) = 
co(%0d) and s(%0d)", vif.a, vif.b, vif.ci, vif.co, vif.s), UVM_MEDIUM);
    end
endtask// End of inlined include file

// You can insert code here by setting monitor_inc_after_class in file 
rca.tpl
```

rca_pkg.sv

```
package rca_pkg;

  `include "uvm_macros.svh"

  import uvm_pkg::*;


  `include "rca_trans.sv"
  `include "rca_config.sv"
  `include "rca_driver.sv"
  `include "rca_monitor.sv"
  `include "rca_sequencer.sv"
  `include "rca_coverage.sv"
  `include "rca_agent.sv"
  `include "rca_seq_lib.sv"

endpackage : rca_pkg
```

rca_seq_lib.sv

```
class rca_default_seq extends uvm_sequence #(trans);

  `uvm_object_utils(rca_default_seq)

  extern function new(string name = "");
  extern task body();

`ifndef UVM_POST_VERSION_1_1
  // Functions to support UVM 1.2 objection API in UVM 1.1
  extern function uvm_phase get_starting_phase();
  extern function void set_starting_phase(uvm_phase phase);
`endif

endclass : rca_default_seq


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


task rca_default_seq::body();
  `uvm_info(get_type_name(), "Default sequence starting", UVM_HIGH)

  req = trans::type_id::create("req");
  start_item(req);
  if ( !req.randomize() )
    `uvm_error(get_type_name(), "Failed to randomize transaction")
  finish_item(req);

  `uvm_info(get_type_name(), "Default sequence completed", UVM_HIGH)
endtask : body


`ifndef UVM_POST_VERSION_1_1
function uvm_phase rca_default_seq::get_starting_phase();
  return starting_phase;
endfunction: get_starting_phase


function void rca_default_seq::set_starting_phase(uvm_phase phase);
  starting_phase = phase;
endfunction: set_starting_phase
`endif


// You can insert code here by setting agent_seq_inc in file rca.tpl
```

rca_sequencer.sv

```
// Sequencer class is specialization of uvm_sequencer
typedef uvm_sequencer #(trans) rca_sequencer_t;
```

rca_trans.sv

```
// You can insert code here by setting trans_inc_before_class in file
rca.tpl

class trans extends uvm_sequence_item;

  `uvm_object_utils(trans)

  // To include variables in copy, compare, print, record, pack, unpack,
and compare2string, define them using trans_var in file rca.tpl
  // To exclude variables from compare, pack, and unpack methods, define
them using trans_meta in file rca.tpl

  // Transaction variables
  rand logic [15:0] input1;
  rand logic [15:0] input2;
  rand logic carryinput;
  logic carryoutput;
  logic [15:0] sum;
  constraint c_addr_a { 0 <= input1; input1 < 5; }
  constraint c_addr_b { 0 <= input2; input2 < 5; }


  extern function new(string name = "");

  // You can remove do_copy/compare/print/record and convert2string
method by setting trans_generate_methods_inside_class = no in file
rca.tpl
  extern function void do_copy(uvm_object rhs);
  extern function bit  do_compare(uvm_object rhs, uvm_comparer
comparer);
  extern function void do_print(uvm_printer printer);
  extern function void do_record(uvm_recorder recorder);
  extern function void do_pack(uvm_packer packer);
  extern function void do_unpack(uvm_packer packer);
  extern function string convert2string();

  // You can insert code here by setting trans_inc_inside_class in file
rca.tpl

endclass : trans


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


// You can remove do_copy/compare/print/record and convert2string method
by setting trans_generate_methods_after_class = no in file rca.tpl

function void trans::do_copy(uvm_object rhs);
  trans rhs_;
  if (!$cast(rhs_, rhs))
```

```
    `uvm_fatal(get_type_name(), "Cast of rhs object failed")
  super.do_copy(rhs);
  input1       = rhs_.input1;
  input2       = rhs_.input2;
  carryinput   = rhs_.carryinput;
  carryoutput  = rhs_.carryoutput;
  sum          = rhs_.sum;
endfunction : do_copy


function bit trans::do_compare(uvm_object rhs, uvm_comparer comparer);
  bit result;
  trans rhs_;
  if (!$cast(rhs_, rhs))
    `uvm_fatal(get_type_name(), "Cast of rhs object failed")
  result = super.do_compare(rhs, comparer);
  result &= comparer.compare_field("input1", input1,
rhs_.input1,      $bits(input1));
  result &= comparer.compare_field("input2", input2,
rhs_.input2,      $bits(input2));
  result &= comparer.compare_field("carryinput", carryinput,
rhs_.carryinput,  $bits(carryinput));
  result &= comparer.compare_field("carryoutput", carryoutput,
rhs_.carryoutput, $bits(carryoutput));
  result &= comparer.compare_field("sum", sum,                 rhs_.sum,
$bits(sum));
  return result;
endfunction : do_compare


function void trans::do_print(uvm_printer printer);
  if (printer.knobs.sprint == 0)
    `uvm_info(get_type_name(), convert2string(), UVM_MEDIUM)
  else
    printer.m_string = convert2string();
endfunction : do_print


function void trans::do_record(uvm_recorder recorder);
  super.do_record(recorder);
  // Use the record macros to record the item fields:
  `uvm_record_field("input1",      input1)
  `uvm_record_field("input2",      input2)
  `uvm_record_field("carryinput",  carryinput)
  `uvm_record_field("carryoutput", carryoutput)
  `uvm_record_field("sum",         sum)
endfunction : do_record


function void trans::do_pack(uvm_packer packer);
  super.do_pack(packer);
  `uvm_pack_int(input1)
  `uvm_pack_int(input2)
  `uvm_pack_int(carryinput)
  `uvm_pack_int(carryoutput)
  `uvm_pack_int(sum)
endfunction : do_pack
```

```
function void trans::do_unpack(uvm_packer packer);
  super.do_unpack(packer);
  `uvm_unpack_int(input1)
  `uvm_unpack_int(input2)
  `uvm_unpack_int(carryinput)
  `uvm_unpack_int(carryoutput)
  `uvm_unpack_int(sum)
endfunction : do_unpack


function string trans::convert2string();
  string s;
  $sformat(s, "%s\n", super.convert2string());
  $sformat(s, {"%s\n",
    "input1      = 'h%0h  'd%0d\n",
    "input2      = 'h%0h  'd%0d\n",
    "carryinput  = 'h%0h  'd%0d\n",
    "carryoutput = 'h%0h  'd%0d\n",
    "sum         = 'h%0h  'd%0d\n"},
    get_full_name(), input1, input1, input2, input2, carryinput,
carryinput, carryoutput, carryoutput, sum, sum);
  return s;
endfunction : convert2string


// You can insert code here by setting trans_inc_after_class in file
rca.tpl
```

10 include

rca_driver_inc.sv

```
task rca_driver::do_drive();
  vif.a <= req.input1;
  vif.b <= req.input2;
  vif.ci <= req.carryinput;
  @(posedge vif.clk);
endtask
```

rca_monitor_inc.sv

```
task rca_monitor::do_mon;
  forever @(posedge vif.clk)
    begin
      m_trans.input1 = vif.a;
      m_trans.input2 = vif.b;
      m_trans.carryinput = vif.ci;
      m_trans.carryoutput = vif.co;
      m_trans.sum = vif.s;
      analysis_port.write(m_trans);
      `uvm_info(get_type_name(),$sformatf("a(%0d) + b(%0d) + ci(%0d) = 
co(%0d) and s(%0d)", vif.a, vif.b, vif.ci, vif.co, vif.s), UVM_MEDIUM);
    end
endtask
```

Appendix A

Simulation Results

```
UVM_INFO ../tb/rca/sv/rca_monitor.sv(71) @ 10000:
uvm_test_top.m_env.m_rca_agent.m_monitor [rca_monitor] a(1) + b(0) + ci(0) =
co(0) and s(1)
UVM_INFO ../tb/rca/sv/rca_monitor.sv(71) @ 30000:
uvm_test_top.m_env.m_rca_agent.m_monitor [rca_monitor] a(4) + b(0) + ci(1) =
co(0) and s(5)
UVM_INFO ../tb/rca/sv/rca_monitor.sv(71) @ 50000:
uvm_test_top.m_env.m_rca_agent.m_monitor [rca_monitor] a(4) + b(4) + ci(0) =
co(0) and s(8)
UVM_INFO ../tb/rca/sv/rca_monitor.sv(71) @ 70000:
uvm_test_top.m_env.m_rca_agent.m_monitor [rca_monitor] a(3) + b(2) + ci(0) =
co(0) and s(5)
UVM_INFO ../tb/rca/sv/rca_monitor.sv(71) @ 90000:
uvm_test_top.m_env.m_rca_agent.m_monitor [rca_monitor] a(0) + b(4) + ci(0) =
co(0) and s(4)
UVM_INFO ../tb/rca/sv/rca_monitor.sv(71) @ 110000:
uvm_test_top.m_env.m_rca_agent.m_monitor [rca_monitor] a(4) + b(1) + ci(1) =
co(0) and s(6)
UVM_INFO ../tb/rca/sv/rca_monitor.sv(71) @ 130000:
uvm_test_top.m_env.m_rca_agent.m_monitor [rca_monitor] a(4) + b(4) + ci(0) =
co(0) and s(8)
UVM_INFO ../tb/rca/sv/rca_monitor.sv(71) @ 150000:
uvm_test_top.m_env.m_rca_agent.m_monitor [rca_monitor] a(1) + b(1) + ci(1) =
co(0) and s(3)
```

Index

1. Doulos. *Easier UVM*. Retrieved from https://www.doulos.com/knowhow/sysverilog/uvm/easier/
2. EDA Playground. *RCA UVM*. Retrieved from https://www.edaplayground.com/x/6HXS