[](http://www.opencores.org/)

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

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.



Figure Easier UVM verification flow.

UVM Architecture Specifications

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



Figure UVM architecture specifications.

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 Easier UVM Code Generator.

## 

Templet Files

### Include File: *rca.tpl*



### Driver File: *rca\_driver\_inc.sv*



### Monitor File: *rca\_monitor\_inc.sv*



### Pin List File: *pinlist*



### Common Templet File: *common.tpl*



### DUT File: *design.sv*



UVM Testbench

**

Figure UVM testbench summary



Figure UVM testbench directory structure

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**

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

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

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

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**



Simulation Results



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