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

LPFFIR Easier UVM

*Author: Vladimir Armstrong*

*vladimirarmstrong@opencores.org*

**Rev. 1.0**

**April 27, 2019**

*This page has been intentionally left blank.*

**Revision History**

| **Rev.** | **Date** | **Author** | **Description** |
| --- | --- | --- | --- |
| 1.0 | 04/27/2019 | Vladimir Armstrong | First Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Contents**

Introduction 1

UVM Architecture Specifications 2

Templet Files 3

common.tpl 3

data\_input.tpl 4

data\_output.tpl 4

pinlist 5

data\_input\_cover\_inc.sv 5

data\_input\_do\_mon.sv 5

data\_input\_driver\_inc\_after\_class.sv 6

data\_input\_driver\_inc\_inside\_class.sv 6

data\_output\_do\_mon.sv 6

reference\_inc\_after\_class.sv 7

reference\_inc\_inside\_class.sv 7

data\_output\_driver\_inc\_inside\_class 7

data\_output\_driver\_inc\_after\_class 8

design.sv 9

Generated UVM Testbench 10

top\_tb 16

top\_tb.sv 16

top\_th.sv 18

top\_test 19

top\_test\_pkg.sv 19

top\_test.sv 20

top 21

port\_converter.sv 21

reference.sv 22

top\_config.sv 24

top\_env.sv 25

top\_pkg.sv 29

top\_seq\_lib.sv 30

data\_input 32

data\_input\_agent.sv 32

data\_input\_config.sv 35

data\_input\_coverage.sv 36

data\_input\_driver.sv 38

data\_input\_if.sv 39

data\_input\_input\_tx.sv 40

data\_input\_monitor.sv 42

data\_input\_pkg.sv 44

data\_input\_seq\_lib.sv 45

data\_input\_sequencer.sv 46

data\_output 47

data\_output\_agent.sv 47

data\_output\_config.sv 50

data\_output\_coverage.sv 51

data\_output\_driver.sv 53

data\_output\_if.sv 54

data\_output\_monitor.sv 55

data\_output\_output\_tx.sv 57

data\_output\_pkg.sv 59

data\_output\_seq\_lib.sv 60

data\_output\_sequencer.sv 61

include 62

data\_input\_cover\_inc.sv 62

data\_input\_do\_mon.sv 63

data\_input\_driver\_inc\_after\_class.sv 64

data\_input\_driver\_inc\_inside\_class.sv 65

data\_output\_do\_mon.sv 66

data\_output\_driver\_inc\_after\_class.sv 67

data\_output\_driver\_inc\_inside\_class.sv 68

reference\_inc\_after\_class.sv 69

reference\_inc\_inside\_class.sv 70

Index 71

Introduction

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



Figure 1 Easier UVM verification flow.

UVM Architecture Specifications



Figure 2 UVM architecture specifications.

## 

Templet Files

### common.tpl

dut\_top **=** lpffir\_axis

nested\_config\_objects **=** yes

tb\_prepend\_to\_initial **=** vcd\_dump**.**sv inline

**#**Path ignored on EDA Playground

**#**syosil\_scoreboard\_src\_path **=** **../../**syosil**/**src

ref\_model\_input **=** reference m\_data\_input\_agent

ref\_model\_output **=** reference m\_data\_output\_agent

ref\_model\_compare\_method **=** reference iop

ref\_model\_inc\_inside\_class **=** reference reference\_inc\_inside\_class**.**sv inline

ref\_model\_inc\_after\_class **=** reference reference\_inc\_after\_class**.**sv inline

top\_default\_seq\_count **=** 10

uvm\_cmdline **=** **+**UVM\_VERBOSITY**=**UVM\_HIGH

### data\_input.tpl

agent\_name **=** data\_input

number\_of\_instances **=** 1

trans\_item **=** input\_tx

trans\_var **=** **rand** **logic** **[**15**:**0**]** data**;**

trans\_var **=** **constraint** c\_data **{** 0 **<=** data**;** data **<** 128**;** **}**

driver\_inc\_inside\_class **=** data\_input\_driver\_inc\_inside\_class**.**sv inline

driver\_inc\_after\_class **=** data\_input\_driver\_inc\_after\_class**.**sv inline

monitor\_inc **=** data\_input\_do\_mon**.**sv inline

agent\_cover\_inc **=** data\_input\_cover\_inc**.**sv inline

if\_port **=** **logic** last**;**

if\_port **=** **logic** valid**;**

if\_port **=** **logic** ready**;**

if\_port **=** **logic** **[**15**:**0**]** data**;**

if\_port **=** **logic** clk**;**

if\_port **=** **logic** reset**;**

if\_clock **=** clk

if\_reset **=** reset

### data\_output.tpl

agent\_name **=** data\_output

number\_of\_instances **=** 1

trans\_item **=** output\_tx

trans\_var **=** **rand** **logic** **[**15**:**0**]** data**;**

agent\_coverage\_enable **=** no

driver\_inc\_inside\_class **=** data\_output\_driver\_inc\_inside\_class**.**sv inline

driver\_inc\_after\_class **=** data\_output\_driver\_inc\_after\_class**.**sv inline

monitor\_inc **=** data\_output\_do\_mon**.**sv inline

if\_port **=** **logic** last**;**

if\_port **=** **logic** valid**;**

if\_port **=** **logic** ready**;**

if\_port **=** **logic** **[**15**:**0**]** data**;**

if\_port **=** **logic** clk**;**

if\_port **=** **logic** reset**;**

if\_clock **=** clk

if\_reset **=** reset

### pinlist

**!**data\_input\_if\_0

rx\_tlast\_i last

rx\_tvalid\_i valid

rx\_tready\_o ready

rx\_tdata\_i data

**!**data\_output\_if\_0

tx\_tlast\_o last

tx\_tvalid\_o valid

tx\_tready\_i ready

tx\_tdata\_o data

**!**

aclk\_i clock

aresetn\_i reset

### data\_input\_cover\_inc.sv

**covergroup** m\_cov**;**

option**.**per\_instance **=** 1**;**

cp\_data**:** **coverpoint** m\_item**.**data **{**

**bins** data\_values**[]** **=** **{[**0**:**127**]};**

**}**

**endgroup**

### data\_input\_do\_mon.sv

**task** data\_input\_monitor**::**do\_mon**;**

**forever** **@(posedge** vif**.**clk**)**

**begin**

**wait** **(**vif**.**reset **==** 1**);**

**if** **(**vif**.**valid **&&** vif**.**ready**)**

**begin**

m\_trans**.**data **=** vif**.**data**;**

analysis\_port**.**write**(**m\_trans**);**

`uvm\_info**(**get\_type\_name**(),** $sformatf**(**"Input data = %0d"**,** m\_trans**.**data**),** UVM\_HIGH**)**

**end**

**end**

**endtask**

### data\_input\_driver\_inc\_after\_class.sv

**task** data\_input\_driver**::**run\_phase**(**uvm\_phase phase**);**

`uvm\_info**(**get\_type\_name**(),** "run\_phase"**,** UVM\_HIGH**)**

**forever** **@(posedge** vif**.**clk**)**

**begin**

seq\_item\_port**.**get\_next\_item**(**req**);**

phase**.**raise\_objection**(this);**

**wait** **(**vif**.**reset **==** 1**);**

vif**.**data **<=** req**.**data**;**

vif**.**valid **<=** 1**;**

vif**.**last **<=** 0**;**

**wait** **(**vif**.**ready **==** 1**);**

**fork**

**begin**

**repeat** **(**10**)** **@(posedge** vif**.**clk**);**

phase**.**drop\_objection**(this);**

**end**

**join\_none**

seq\_item\_port**.**item\_done**();**

**end**

**endtask** **:** run\_phase

### data\_input\_driver\_inc\_inside\_class.sv

**extern** **task** run\_phase**(**uvm\_phase phase**);**

### data\_output\_do\_mon.sv

**task** data\_output\_monitor**::**do\_mon**;**

**forever** **@(posedge** vif**.**clk**)**

**begin**

**wait** **(**vif**.**reset **==** 1**);**

**if** **(**vif**.**valid **&&** vif**.**ready**)**

**begin**

m\_trans**.**data **=** vif**.**data**;**

analysis\_port**.**write**(**m\_trans**);**

`uvm\_info**(**get\_type\_name**(),** $sformatf**(**"Output data = %0d"**,**m\_trans**.**data**),** UVM\_HIGH**)**

**end**

**end**

**endtask**

### reference\_inc\_after\_class.sv

**function** **void** reference**::**write\_reference\_0**(**input\_tx t**);**

send**(**t**);**

**endfunction**

**function** **void** reference**::**send**(**input\_tx t**);**

output\_tx tx**;**

tx **=** output\_tx**::**type\_id**::**create**(**"tx"**);**

**if** **(**init\_flag **==** 1**)**

**begin**

init\_flag **=** 0**;**

**foreach(**tx\_save**[**j**])**

tx\_save**[**j**]** **=** 0**;**

**end**

**if** **(**save\_pnt **==** 5**)**

save\_pnt **=** 0**;**

**else**

save\_pnt**++;**

tx\_save**[**save\_pnt**]** **=** t**.**data**;**

tx**.**data **=** tx\_save**[**0**]** **+** tx\_save**[**1**]** **+** tx\_save**[**2**]** **+** tx\_save**[**3**]** **+** tx\_save**[**4**]** **+** tx\_save**[**5**];**

analysis\_port\_0**.**write**(**tx**);**

`uvm\_info**(**get\_type\_name**(),** $sformatf**(**"Reference Model save\_pnt = %0d, data = %0d"**,**save\_pnt**,** tx**.**data**),** UVM\_HIGH**)**

**endfunction**

### reference\_inc\_inside\_class.sv

**extern** **function** **void** send**(**input\_tx t**);**

**int** save\_pnt **=** 5**;**

**logic** **[**15**:**0**]** tx\_save **[**0**:**5**];**

**int** init\_flag **=** 1**;**

### data\_output\_driver\_inc\_inside\_class

**extern** **task** run\_phase**(**uvm\_phase phase**);**

### data\_output\_driver\_inc\_after\_class

**task** data\_output\_driver**::**run\_phase**(**uvm\_phase phase**);**

`uvm\_info**(**get\_type\_name**(),** "run\_phase"**,** UVM\_HIGH**)**

**forever** **@(posedge** vif**.**clk**)**

**begin**

seq\_item\_port**.**get\_next\_item**(**req**);**

phase**.**raise\_objection**(this);**

vif**.**ready **<=** 1**;**

**wait** **(**vif**.**reset **==** 1**);**

**fork**

**begin**

**repeat** **(**10**)** **@(posedge** vif**.**clk**);**

phase**.**drop\_objection**(this);**

**end**

**join\_none**

seq\_item\_port**.**item\_done**();**

**end**

**endtask** **:** run\_phase

### design.sv

**module** lpffir\_axis **(**

**input** aclk\_i**,**

**input** aresetn\_i**,**

// AXI-Stream RX interface

**input** rx\_tlast\_i**,**

**input** rx\_tvalid\_i**,**

**output** **logic** rx\_tready\_o**,**

**input** **[**15**:**0**]** rx\_tdata\_i**,**

// AXI-Stream TX interface

**output** **logic** tx\_tlast\_o**,**

**output** **reg** tx\_tvalid\_o**,**

**input** tx\_tready\_i**,**

**output** **logic** **[**15**:**0**]** tx\_tdata\_o

**);**

**wire** lpffir\_en**;**

**assign** lpffir\_en **=** rx\_tvalid\_i **&&** tx\_tready\_i**;**

// AXI-Stream interface

**assign** rx\_tready\_o **=** lpffir\_en**;**

**assign** tx\_tvalid\_o **=** lpffir\_en**;**

**assign** tx\_tlast\_o **=** rx\_tlast\_i**;**

// DEBUG

**always** **@(posedge** aclk\_i **or** **negedge** aresetn\_i**)**

**if** **(**aresetn\_i**)**

$display**(**"DUT: rx\_tdata\_i %0d, tx\_tdata\_o %0d"**,** rx\_tdata\_i**,** tx\_tdata\_o**);**

// LPFFIR

lpffir\_core lpffir\_core**(**

**.**clk\_i**(**aclk\_i**),**

**.**rstn\_i**(**aresetn\_i**),**

**.**en\_i**(**lpffir\_en**),**

**.**x\_i**(**rx\_tdata\_i**),**

**.**y\_o**(**tx\_tdata\_o**)**

**);**

**endmodule**

Generated UVM Testbench

--------------------------------------------------------------------------------

Name Type Size Value

--------------------------------------------------------------------------------

uvm\_test\_top top\_test - @347

m\_env top\_env - @360

m\_converter\_m\_data\_output\_agent uvm\_component - @417

a\_port uvm\_analysis\_port - @436

analysis\_imp uvm\_analysis\_imp - @426

m\_data\_input\_agent data\_input\_agent - @455

analysis\_port uvm\_analysis\_port - @464

m\_driver data\_input\_driver - @552

rsp\_port uvm\_analysis\_port - @571

seq\_item\_port uvm\_seq\_item\_pull\_port - @561

m\_monitor data\_input\_monitor - @532

analysis\_port uvm\_analysis\_port - @541

m\_sequencer uvm\_sequencer - @581

rsp\_export uvm\_analysis\_export - @590

seq\_item\_export uvm\_seq\_item\_pull\_imp - @708

arbitration\_queue array 0 -

lock\_queue array 0 -

num\_last\_reqs integral 32 'd1

num\_last\_rsps integral 32 'd1

m\_data\_input\_coverage data\_input\_coverage - @474

analysis\_imp uvm\_analysis\_imp - @483

m\_data\_output\_agent data\_output\_agent - @493

analysis\_port uvm\_analysis\_port - @502

m\_driver data\_output\_driver - @745

rsp\_port uvm\_analysis\_port - @764

seq\_item\_port uvm\_seq\_item\_pull\_port - @754

m\_monitor data\_output\_monitor - @725

analysis\_port uvm\_analysis\_port - @734

m\_sequencer uvm\_sequencer - @774

rsp\_export uvm\_analysis\_export - @783

seq\_item\_export uvm\_seq\_item\_pull\_imp - @901

arbitration\_queue array 0 -

lock\_queue array 0 -

num\_last\_reqs integral 32 'd1

num\_last\_rsps integral 32 'd1

m\_data\_output\_coverage data\_output\_coverage - @512

analysis\_imp uvm\_analysis\_imp - @521

m\_reference reference - @388

analysis\_export\_0 uvm\_analysis\_imp\_reference\_0 - @397

analysis\_port\_0 uvm\_analysis\_port - @407

m\_reference\_scoreboard cl\_syoscb - @446

DUT cl\_syoscb\_queue\_std - @918

cfg cl\_syoscb\_cfg - @382

queues aa(object,string) 2 -

[DUT] cl\_syoscb\_queue\_std - @918

[REF] cl\_syoscb\_queue\_std - @929

cfg cl\_syoscb\_cfg - @382

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

producers aa(object,string) 1 -

[m\_data\_output\_agent] cl\_syoscb\_cfg\_pl - @383

list da(string) 2 -

[0] string 3 DUT

[1] string 3 REF

primary\_queue string 3 DUT

disable\_clone integral 1 'h0

max\_queue\_size aa(int,string) 2 -

[DUT] integral 32 'h0

[REF] integral 32 'h0

scb\_name string 22 m\_reference\_scoreboard

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

REF cl\_syoscb\_queue\_std - @929

cfg cl\_syoscb\_cfg - @382

queues aa(object,string) 2 -

[DUT] cl\_syoscb\_queue\_std - @918

cfg cl\_syoscb\_cfg - @382

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

[REF] cl\_syoscb\_queue\_std - @929

producers aa(object,string) 1 -

[m\_data\_output\_agent] cl\_syoscb\_cfg\_pl - @383

list da(string) 2 -

[0] string 3 DUT

[1] string 3 REF

primary\_queue string 3 DUT

disable\_clone integral 1 'h0

max\_queue\_size aa(int,string) 2 -

[DUT] integral 32 'h0

[REF] integral 32 'h0

scb\_name string 22 m\_reference\_scoreboard

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

compare\_strategy cl\_syoscb\_compare - @940

cfg cl\_syoscb\_cfg - @382

queues aa(object,string) 2 -

[DUT] cl\_syoscb\_queue\_std - @918

cfg cl\_syoscb\_cfg - @382

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

[REF] cl\_syoscb\_queue\_std - @929

cfg cl\_syoscb\_cfg - @382

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

producers aa(object,string) 1 -

[m\_data\_output\_agent] cl\_syoscb\_cfg\_pl - @383

list da(string) 2 -

[0] string 3 DUT

[1] string 3 REF

primary\_queue string 3 DUT

disable\_clone integral 1 'h0

max\_queue\_size aa(int,string) 2 -

[DUT] integral 32 'h0

[REF] integral 32 'h0

scb\_name string 22 m\_reference\_scoreboard

compare\_algo cl\_syoscb\_compare\_iop - @991

cfg cl\_syoscb\_cfg - @382

queues aa(object,string) 2 -

[DUT] cl\_syoscb\_queue\_std - @918

cfg cl\_syoscb\_cfg - @382

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

[REF] cl\_syoscb\_queue\_std - @929

cfg cl\_syoscb\_cfg - @382

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

producers aa(object,string) 1 -

[m\_data\_output\_agent] cl\_syoscb\_cfg\_pl - @383

list da(string) 2 -

[0] string 3 DUT

[1] string 3 REF

primary\_queue string 3 DUT

disable\_clone integral 1 'h0

max\_queue\_size aa(int,string) 2 -

[DUT] integral 32 'h0

[REF] integral 32 'h0

scb\_name string 22 m\_reference\_scoreboard

m\_data\_output\_agent\_DUT\_subscr cl\_syoscb\_subscriber - @950

analysis\_imp uvm\_analysis\_imp - @959

queue\_name string 3 DUT

producer string 19 m\_data\_output\_agent

m\_data\_output\_agent\_REF\_subscr cl\_syoscb\_subscriber - @969

analysis\_imp uvm\_analysis\_imp - @978

queue\_name string 3 REF

producer string 19 m\_data\_output\_agent

cfg cl\_syoscb\_cfg - @382

queues aa(object,string) 2 -

[DUT] cl\_syoscb\_queue\_std - @918

cfg cl\_syoscb\_cfg - @382

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

[REF] cl\_syoscb\_queue\_std - @929

cfg cl\_syoscb\_cfg - @382

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

producers aa(object,string) 1 -

[m\_data\_output\_agent] cl\_syoscb\_cfg\_pl - @383

list da(string) 2 -

[0] string 3 DUT

[1] string 3 REF

primary\_queue string 3 DUT

disable\_clone integral 1 'h0

max\_queue\_size aa(int,string) 2 -

[DUT] integral 32 'h0

[REF] integral 32 'h0

scb\_name string 22 m\_reference\_scoreboard

queues da(object) 2 -

[0] cl\_syoscb\_queue\_std - @918

cfg cl\_syoscb\_cfg - @382

queues aa(object,string) 2 -

[DUT] cl\_syoscb\_queue\_std - @918

[REF] cl\_syoscb\_queue\_std - @929

cfg cl\_syoscb\_cfg - @382

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

producers aa(object,string) 1 -

[m\_data\_output\_agent] cl\_syoscb\_cfg\_pl - @383

list da(string) 2 -

[0] string 3 DUT

[1] string 3 REF

primary\_queue string 3 DUT

disable\_clone integral 1 'h0

max\_queue\_size aa(int,string) 2 -

[DUT] integral 32 'h0

[REF] integral 32 'h0

scb\_name string 22 m\_reference\_scoreboard

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

[1] cl\_syoscb\_queue\_std - @929

cfg cl\_syoscb\_cfg - @382

queues aa(object,string) 2 -

[DUT] cl\_syoscb\_queue\_std - @918

cfg cl\_syoscb\_cfg - @382

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

[REF] cl\_syoscb\_queue\_std - @929

producers aa(object,string) 1 -

[m\_data\_output\_agent] cl\_syoscb\_cfg\_pl - @383

list da(string) 2 -

[0] string 3 DUT

[1] string 3 REF

primary\_queue string 3 DUT

disable\_clone integral 1 'h0

max\_queue\_size aa(int,string) 2 -

[DUT] integral 32 'h0

[REF] integral 32 'h0

scb\_name string 22 m\_reference\_scoreboard

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

compare\_strategy cl\_syoscb\_compare - @940

cfg cl\_syoscb\_cfg - @382

queues aa(object,string) 2 -

[DUT] cl\_syoscb\_queue\_std - @918

cfg cl\_syoscb\_cfg - @382

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

[REF] cl\_syoscb\_queue\_std - @929

cfg cl\_syoscb\_cfg - @382

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

producers aa(object,string) 1 -

[m\_data\_output\_agent] cl\_syoscb\_cfg\_pl - @383

list da(string) 2 -

[0] string 3 DUT

[1] string 3 REF

primary\_queue string 3 DUT

disable\_clone integral 1 'h0

max\_queue\_size aa(int,string) 2 -

[DUT] integral 32 'h0

[REF] integral 32 'h0

scb\_name string 22 m\_reference\_scoreboard

compare\_algo cl\_syoscb\_compare\_iop - @991

cfg cl\_syoscb\_cfg - @382

queues aa(object,string) 2 -

[DUT] cl\_syoscb\_queue\_std - @918

cfg cl\_syoscb\_cfg - @382

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

[REF] cl\_syoscb\_queue\_std - @929

cfg cl\_syoscb\_cfg - @382

iter\_idx integral 32 'h0

cnt\_add\_item integral 32 'h0

items da(object) 0 -

producers aa(object,string) 1 -

[m\_data\_output\_agent] cl\_syoscb\_cfg\_pl - @383

list da(string) 2 -

[0] string 3 DUT

[1] string 3 REF

primary\_queue string 3 DUT

disable\_clone integral 1 'h0

max\_queue\_size aa(int,string) 2 -

[DUT] integral 32 'h0

[REF] integral 32 'h0

scb\_name string 22 m\_reference\_scoreboard

subscribers aa(object,string) 2 -

[DUTm\_data\_output\_agent] cl\_syoscb\_subscriber - @950

analysis\_imp uvm\_analysis\_imp - @959

queue\_name string 3 DUT

producer string 19 m\_data\_output\_agent

[REFm\_data\_output\_agent] cl\_syoscb\_subscriber - @969

analysis\_imp uvm\_analysis\_imp - @978

queue\_name string 3 REF

producer string 19 m\_data\_output\_agent

--------------------------------------------------------------------------------

Figure 3 UVM testbench topology

tb

├── data\_input

│ └── sv

│ ├── data\_input\_agent.sv

│ ├── data\_input\_config.sv

│ ├── data\_input\_coverage.sv

│ ├── data\_input\_driver.sv

│ ├── data\_input\_if.sv

│ ├── data\_input\_input\_tx.sv

│ ├── data\_input\_monitor.sv

│ ├── data\_input\_pkg.sv

│ ├── data\_input\_seq\_lib.sv

│ └── data\_input\_sequencer.sv

├── data\_output

│ └── sv

│ ├── data\_output\_agent.sv

│ ├── data\_output\_config.sv

│ ├── data\_output\_coverage.sv

│ ├── data\_output\_driver.sv

│ ├── data\_output\_if.sv

│ ├── data\_output\_monitor.sv

│ ├── data\_output\_output\_tx.sv

│ ├── data\_output\_pkg.sv

│ ├── data\_output\_seq\_lib.sv

│ └── data\_output\_sequencer.sv

├── include

│ ├── data\_input\_cover\_inc.sv

│ ├── data\_input\_do\_mon.sv

│ ├── data\_input\_driver\_inc\_after\_class.sv

│ ├── data\_input\_driver\_inc\_inside\_class.sv

│ ├── data\_output\_do\_mon.sv

│ ├── data\_output\_driver\_inc\_after\_class.sv

│ ├── data\_output\_driver\_inc\_inside\_class.sv

│ ├── reference\_inc\_after\_class.sv

│ ├── reference\_inc\_inside\_class.sv

│ └── vcd\_dump.sv

├── top

│ └── sv

│ ├── port\_converter.sv

│ ├── reference.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 4 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**

// Start of inlined include file generated\_tb/tb/include/vcd\_dump.sv

$dumpfile**(**"dump.vcd"**);**

$dumpvars**;**

// End of inlined include file

// 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**.**m\_data\_input\_config**.**vif **=** th**.**data\_input\_if\_0**;**

top\_env\_config**.**m\_data\_output\_config**.**vif **=** th**.**data\_output\_if\_0**;**

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** data\_input\_if\_0**.**reset **=** reset**;**

**assign** data\_output\_if\_0**.**reset **=** reset**;**

**assign** data\_input\_if\_0**.**clk **=** clock**;**

**assign** data\_output\_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

data\_input\_if data\_input\_if\_0 **();**

data\_output\_if data\_output\_if\_0 **();**

lpffir\_axis uut **(**

**.**rx\_tlast\_i **(**data\_input\_if\_0**.**last**),**

**.**rx\_tvalid\_i**(**data\_input\_if\_0**.**valid**),**

**.**rx\_tready\_o**(**data\_input\_if\_0**.**ready**),**

**.**rx\_tdata\_i **(**data\_input\_if\_0**.**data**),**

**.**tx\_tlast\_o **(**data\_output\_if\_0**.**last**),**

**.**tx\_tvalid\_o**(**data\_output\_if\_0**.**valid**),**

**.**tx\_tready\_i**(**data\_output\_if\_0**.**ready**),**

**.**tx\_tdata\_o **(**data\_output\_if\_0**.**data**),**

**.**aclk\_i **(**clock**),**

**.**aresetn\_i **(**reset**)**

**);**

**endmodule**

top\_test

### top\_test\_pkg.sv

**package** top\_test\_pkg**;**

`include "uvm\_macros.svh"

**import** uvm\_pkg**::\*;**

**import** data\_input\_pkg**::\*;**

**import** data\_output\_pkg**::\*;**

**import** top\_pkg**::\*;**

`include "top\_test.sv"

**endpackage** **:** top\_test\_pkg

### top\_test.sv

**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

top

### port\_converter.sv

**class** port\_converter **#(type** T **=** uvm\_sequence\_item**)** **extends** uvm\_subscriber **#(**T**);**

`uvm\_component\_param\_utils**(**port\_converter**#(**T**))**

// For connecting analysis port of monitor to analysis export of Syosil scoreboard

uvm\_analysis\_port **#(**uvm\_sequence\_item**)** analysis\_port**;**

**function** **new(string** name**,** uvm\_component parent**);**

**super.**new**(**name**,** parent**);**

analysis\_port **=** **new(**"a\_port"**,** **this);**

**endfunction**

**function** **void** write**(**T t**);**

analysis\_port**.**write**(**t**);**

**endfunction**

**endclass**

### reference.sv

`uvm\_analysis\_imp\_decl**(**\_reference\_0**)**

**class** reference **extends** uvm\_component**;**

`uvm\_component\_utils**(**reference**)**

uvm\_analysis\_imp\_reference\_0 **#(**input\_tx**,** reference**)** analysis\_export\_0**;** // m\_data\_input\_agent

uvm\_analysis\_port **#(**uvm\_sequence\_item**)** analysis\_port\_0**;** // m\_data\_output\_agent

**extern** **function** **new(string** name**,** uvm\_component parent**);**

**extern** **function** **void** write\_reference\_0**(input** input\_tx t**);**

// Start of inlined include file generated\_tb/tb/include/reference\_inc\_inside\_class.sv

**extern** **function** **void** send**(**input\_tx t**);**

**int** save\_pnt **=** 5**;**

**logic** **[**15**:**0**]** tx\_save **[**0**:**5**];**

**int** init\_flag **=** 1**;** // End of inlined include file

**endclass**

**function** reference**::new(string** name**,** uvm\_component parent**);**

**super.**new**(**name**,** parent**);**

analysis\_export\_0 **=** **new(**"analysis\_export\_0"**,** **this);**

analysis\_port\_0 **=** **new(**"analysis\_port\_0"**,** **this);**

**endfunction** **:** **new**

// Start of inlined include file generated\_tb/tb/include/reference\_inc\_after\_class.sv

**function** **void** reference**::**write\_reference\_0**(**input\_tx t**);**

send**(**t**);**

**endfunction**

**function** **void** reference**::**send**(**input\_tx t**);**

output\_tx tx**;**

tx **=** output\_tx**::**type\_id**::**create**(**"tx"**);**

**if** **(**init\_flag **==** 1**)**

**begin**

init\_flag **=** 0**;**

**foreach(**tx\_save**[**j**])**

tx\_save**[**j**]** **=** 0**;**

**end**

**if** **(**save\_pnt **==** 5**)**

save\_pnt **=** 0**;**

**else**

save\_pnt**++;**

tx\_save**[**save\_pnt**]** **=** t**.**data**;**

tx**.**data **=** tx\_save**[**0**]** **+** tx\_save**[**1**]** **+** tx\_save**[**2**]** **+** tx\_save**[**3**]** **+** tx\_save**[**4**]** **+** tx\_save**[**5**];**

analysis\_port\_0**.**write**(**tx**);**

`uvm\_info**(**get\_type\_name**(),** $sformatf**(**"Reference Model save\_pnt = %0d, data = %0d"**,**save\_pnt**,** tx**.**data**),** UVM\_HIGH**)**

**endfunction**

### top\_config.sv

**class** top\_config **extends** uvm\_object**;**

// Do not register config class with the factory

**rand** data\_input\_config m\_data\_input\_config**;**

**rand** data\_output\_config m\_data\_output\_config**;**

// 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**);**

m\_data\_input\_config **=** **new(**"m\_data\_input\_config"**);**

m\_data\_input\_config**.**is\_active **=** UVM\_ACTIVE**;**

m\_data\_input\_config**.**checks\_enable **=** 1**;**

m\_data\_input\_config**.**coverage\_enable **=** 1**;**

m\_data\_output\_config **=** **new(**"m\_data\_output\_config"**);**

m\_data\_output\_config**.**is\_active **=** UVM\_ACTIVE**;**

m\_data\_output\_config**.**checks\_enable **=** 1**;**

m\_data\_output\_config**.**coverage\_enable **=** 0**;**

// You can insert code here by setting top\_env\_config\_append\_to\_new in file common.tpl

**endfunction** **:** **new**

### top\_env.sv

**import** pk\_syoscb**::\*;**

**class** top\_env **extends** uvm\_env**;**

`uvm\_component\_utils**(**top\_env**)**

**extern** **function** **new(string** name**,** uvm\_component parent**);**

// Reference model and Syosil scoreboard

**typedef** port\_converter **#(**output\_tx**)** converter\_m\_data\_output\_agent\_t**;**

converter\_m\_data\_output\_agent\_t m\_converter\_m\_data\_output\_agent**;**

reference m\_reference**;**

cl\_syoscb m\_reference\_scoreboard**;**

cl\_syoscb\_cfg m\_reference\_config**;**

// Child agents

data\_input\_config m\_data\_input\_config**;**

data\_input\_agent m\_data\_input\_agent**;**

data\_input\_coverage m\_data\_input\_coverage**;**

data\_output\_config m\_data\_output\_config**;**

data\_output\_agent m\_data\_output\_agent**;**

data\_output\_coverage m\_data\_output\_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\_data\_input\_config **=** m\_config**.**m\_data\_input\_config**;**

// You can insert code here by setting agent\_copy\_config\_vars in file data\_input.tpl

uvm\_config\_db **#(**data\_input\_config**)::**set**(this,** "m\_data\_input\_agent"**,** "config"**,** m\_data\_input\_config**);**

**if** **(**m\_data\_input\_config**.**is\_active **==** UVM\_ACTIVE **)**

uvm\_config\_db **#(**data\_input\_config**)::**set**(this,** "m\_data\_input\_agent.m\_sequencer"**,** "config"**,** m\_data\_input\_config**);**

uvm\_config\_db **#(**data\_input\_config**)::**set**(this,** "m\_data\_input\_coverage"**,** "config"**,** m\_data\_input\_config**);**

m\_data\_output\_config **=** m\_config**.**m\_data\_output\_config**;**

// You can insert code here by setting agent\_copy\_config\_vars in file data\_output.tpl

uvm\_config\_db **#(**data\_output\_config**)::**set**(this,** "m\_data\_output\_agent"**,** "config"**,** m\_data\_output\_config**);**

**if** **(**m\_data\_output\_config**.**is\_active **==** UVM\_ACTIVE **)**

uvm\_config\_db **#(**data\_output\_config**)::**set**(this,** "m\_data\_output\_agent.m\_sequencer"**,** "config"**,** m\_data\_output\_config**);**

uvm\_config\_db **#(**data\_output\_config**)::**set**(this,** "m\_data\_output\_coverage"**,** "config"**,** m\_data\_output\_config**);**

// Default factory overrides for Syosil scoreboard

cl\_syoscb\_queue**::**type\_id**::**set\_type\_override**(**cl\_syoscb\_queue\_std**::**type\_id**::**get**());**

**begin**

**bit** ok**;**

uvm\_factory factory **=** uvm\_factory**::**get**();**

**if** **(**factory**.**find\_override\_by\_type**(**cl\_syoscb\_compare\_base**::**type\_id**::**get**(),** "\*"**)** **==** cl\_syoscb\_compare\_base**::**type\_id**::**get**())**

cl\_syoscb\_compare\_base**::**type\_id**::**set\_inst\_override**(**cl\_syoscb\_compare\_iop**::**type\_id**::**get**(),** "m\_reference\_scoreboard.\*"**,** **this);**

// Configuration object for Syosil scoreboard

m\_reference\_config **=** cl\_syoscb\_cfg**::**type\_id**::**create**(**"m\_reference\_config"**);**

m\_reference\_config**.**set\_queues**(** **{**"DUT"**,** "REF"**}** **);**

ok **=** m\_reference\_config**.**set\_primary\_queue**(**"DUT"**);**

**assert(**ok**);**

ok **=** m\_reference\_config**.**set\_producer**(**"m\_data\_output\_agent"**,** **{**"DUT"**,** "REF"**}** **);**

**assert(**ok**);**

uvm\_config\_db**#(**cl\_syoscb\_cfg**)::**set**(this,** "m\_reference\_scoreboard"**,** "cfg"**,** m\_reference\_config**);**

// Instantiate reference model and Syosil scoreboard

m\_reference **=** reference **::**type\_id**::**create**(**"m\_reference"**,** **this);**

m\_converter\_m\_data\_output\_agent **=** converter\_m\_data\_output\_agent\_t**::**type\_id**::**create**(**"m\_converter\_m\_data\_output\_agent"**,** **this);**

m\_reference\_scoreboard **=** cl\_syoscb **::**type\_id**::**create**(**"m\_reference\_scoreboard"**,** **this);**

**end**

m\_data\_input\_agent **=** data\_input\_agent **::**type\_id**::**create**(**"m\_data\_input\_agent"**,** **this);**

m\_data\_input\_coverage **=** data\_input\_coverage **::**type\_id**::**create**(**"m\_data\_input\_coverage"**,** **this);**

m\_data\_output\_agent **=** data\_output\_agent **::**type\_id**::**create**(**"m\_data\_output\_agent"**,** **this);**

m\_data\_output\_coverage **=** data\_output\_coverage**::**type\_id**::**create**(**"m\_data\_output\_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\_data\_input\_agent**.**analysis\_port**.**connect**(**m\_data\_input\_coverage**.**analysis\_export**);**

m\_data\_output\_agent**.**analysis\_port**.**connect**(**m\_data\_output\_coverage**.**analysis\_export**);**

**begin**

// Connect reference model and Syosil scoreboard

cl\_syoscb\_subscriber subscriber**;**

m\_data\_input\_agent**.**analysis\_port**.**connect**(**m\_reference**.**analysis\_export\_0**);**

subscriber **=** m\_reference\_scoreboard**.**get\_subscriber**(**"REF"**,** "m\_data\_output\_agent"**);**

m\_reference**.**analysis\_port\_0**.**connect**(**subscriber**.**analysis\_export**);**

subscriber **=** m\_reference\_scoreboard**.**get\_subscriber**(**"DUT"**,** "m\_data\_output\_agent"**);**

m\_data\_output\_agent**.**analysis\_port**.**connect**(**m\_converter\_m\_data\_output\_agent**.**analysis\_export**);**

m\_converter\_m\_data\_output\_agent**.**analysis\_port**.**connect**(**subscriber**.**analysis\_export**);**

**end**

// 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\_data\_input\_agent **=** m\_data\_input\_agent**;**

vseq**.**m\_data\_output\_agent **=** m\_data\_output\_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

### top\_pkg.sv

**package** top\_pkg**;**

`include "uvm\_macros.svh"

**import** uvm\_pkg**::\*;**

**import** data\_input\_pkg**::\*;**

**import** data\_output\_pkg**::\*;**

`include "top\_config.sv"

`include "top\_seq\_lib.sv"

`include "port\_converter.sv"

`include "reference.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**)**

data\_input\_agent m\_data\_input\_agent**;**

data\_output\_agent m\_data\_output\_agent**;**

// Number of times to repeat child sequences

**int** m\_seq\_count **=** 10**;**

**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\_data\_input\_agent**.**m\_config**.**is\_active **==** UVM\_ACTIVE**)**

**begin**

data\_input\_default\_seq seq**;**

seq **=** data\_input\_default\_seq**::**type\_id**::**create**(**"seq"**);**

seq**.**set\_item\_context**(this,** m\_data\_input\_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\_data\_input\_agent**.**m\_sequencer**,** **this);**

**end**

**if** **(**m\_data\_output\_agent**.**m\_config**.**is\_active **==** UVM\_ACTIVE**)**

**begin**

data\_output\_default\_seq seq**;**

seq **=** data\_output\_default\_seq**::**type\_id**::**create**(**"seq"**);**

seq**.**set\_item\_context**(this,** m\_data\_output\_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\_data\_output\_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

data\_input

### data\_input\_agent.sv

**class** data\_input\_agent **extends** uvm\_agent**;**

`uvm\_component\_utils**(**data\_input\_agent**)**

uvm\_analysis\_port **#(**input\_tx**)** analysis\_port**;**

data\_input\_config m\_config**;**

data\_input\_sequencer\_t m\_sequencer**;**

data\_input\_driver m\_driver**;**

data\_input\_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 data\_input.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 data\_input.tpl

**endclass** **:** data\_input\_agent

**function** data\_input\_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 data\_input.tpl

**function** **void** data\_input\_agent**::**build\_phase**(**uvm\_phase phase**);**

// You can insert code here by setting agent\_prepend\_to\_build\_phase in file data\_input.tpl

**if** **(!**uvm\_config\_db **#(**data\_input\_config**)::**get**(this,** ""**,** "config"**,** m\_config**))**

`uvm\_error**(**get\_type\_name**(),** "data\_input config not found"**)**

m\_monitor **=** data\_input\_monitor **::**type\_id**::**create**(**"m\_monitor"**,** **this);**

**if** **(**get\_is\_active**()** **==** UVM\_ACTIVE**)**

**begin**

m\_driver **=** data\_input\_driver **::**type\_id**::**create**(**"m\_driver"**,** **this);**

m\_sequencer **=** data\_input\_sequencer\_t**::**type\_id**::**create**(**"m\_sequencer"**,** **this);**

**end**

// You can insert code here by setting agent\_append\_to\_build\_phase in file data\_input.tpl

**endfunction** **:** build\_phase

**function** **void** data\_input\_agent**::**connect\_phase**(**uvm\_phase phase**);**

**if** **(**m\_config**.**vif **==** **null)**

`uvm\_warning**(**get\_type\_name**(),** "data\_input 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 data\_input.tpl

**endfunction** **:** connect\_phase

**function** uvm\_active\_passive\_enum data\_input\_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

### data\_input\_config.sv

**class** data\_input\_config **extends** uvm\_object**;**

// Do not register config class with the factory

**virtual** data\_input\_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 data\_input.tpl

// You can remove new by setting agent\_config\_generate\_methods\_inside\_class = no in file data\_input.tpl

**extern** **function** **new(string** name **=** ""**);**

// You can insert code here by setting agent\_config\_inc\_inside\_class in file data\_input.tpl

**endclass** **:** data\_input\_config

// You can remove new by setting agent\_config\_generate\_methods\_after\_class = no in file data\_input.tpl

**function** data\_input\_config**::new(string** name **=** ""**);**

**super.**new**(**name**);**

**endfunction** **:** **new**

### data\_input\_coverage.sv

**class** data\_input\_coverage **extends** uvm\_subscriber **#(**input\_tx**);**

`uvm\_component\_utils**(**data\_input\_coverage**)**

data\_input\_config m\_config**;**

**bit** m\_is\_covered**;**

input\_tx m\_item**;**

// Start of inlined include file generated\_tb/tb/include/data\_input\_cover\_inc.sv

**covergroup** m\_cov**;**

option**.**per\_instance **=** 1**;**

cp\_data**:** **coverpoint** m\_item**.**data **{**

**bins** data\_values**[]** **=** **{[**0**:**127**]};**

**}**

**endgroup**

// End of inlined include file

// You can remove new, write, and report\_phase by setting agent\_cover\_generate\_methods\_inside\_class = no in file data\_input.tpl

**extern** **function** **new(string** name**,** uvm\_component parent**);**

**extern** **function** **void** write**(input** input\_tx 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 data\_input.tpl

**endclass** **:** data\_input\_coverage

// You can remove new, write, and report\_phase by setting agent\_cover\_generate\_methods\_after\_class = no in file data\_input.tpl

**function** data\_input\_coverage**::new(string** name**,** uvm\_component parent**);**

**super.**new**(**name**,** parent**);**

m\_is\_covered **=** 0**;**

m\_cov **=** **new();**

**endfunction** **:** **new**

**function** **void** data\_input\_coverage**::**write**(input** input\_tx 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** data\_input\_coverage**::**build\_phase**(**uvm\_phase phase**);**

**if** **(!**uvm\_config\_db **#(**data\_input\_config**)::**get**(this,** ""**,** "config"**,** m\_config**))**

`uvm\_error**(**get\_type\_name**(),** "data\_input config not found"**)**

**endfunction** **:** build\_phase

**function** **void** data\_input\_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

### data\_input\_driver.sv

**class** data\_input\_driver **extends** uvm\_driver **#(**input\_tx**);**

`uvm\_component\_utils**(**data\_input\_driver**)**

**virtual** data\_input\_if vif**;**

**extern** **function** **new(string** name**,** uvm\_component parent**);**

// Start of inlined include file generated\_tb/tb/include/data\_input\_driver\_inc\_inside\_class.sv

**extern** **task** run\_phase**(**uvm\_phase phase**);**

// End of inlined include file

**endclass** **:** data\_input\_driver

**function** data\_input\_driver**::new(string** name**,** uvm\_component parent**);**

**super.**new**(**name**,** parent**);**

**endfunction** **:** **new**

// Start of inlined include file generated\_tb/tb/include/data\_input\_driver\_inc\_after\_class.sv

**task** data\_input\_driver**::**run\_phase**(**uvm\_phase phase**);**

`uvm\_info**(**get\_type\_name**(),** "run\_phase"**,** UVM\_HIGH**)**

**forever** **@(posedge** vif**.**clk**)**

**begin**

seq\_item\_port**.**get\_next\_item**(**req**);**

phase**.**raise\_objection**(this);**

**wait** **(**vif**.**reset **==** 1**);**

vif**.**data **<=** req**.**data**;**

vif**.**valid **<=** 1**;**

vif**.**last **<=** 0**;**

**wait** **(**vif**.**ready **==** 1**);**

**fork**

**begin**

**repeat** **(**10**)** **@(posedge** vif**.**clk**);**

phase**.**drop\_objection**(this);**

**end**

**join\_none**

seq\_item\_port**.**item\_done**();**

**end**

**endtask** **:** run\_phase

### data\_input\_if.sv

**interface** data\_input\_if**();**

**timeunit** 1ns**;**

**timeprecision** 1ps**;**

**import** data\_input\_pkg**::\*;**

**logic** last**;**

**logic** valid**;**

**logic** ready**;**

**logic** **[**15**:**0**]** data**;**

**logic** clk**;**

**logic** reset**;**

// You can insert properties and assertions here

// You can insert code here by setting if\_inc\_inside\_interface in file data\_input.tpl

**endinterface** **:** data\_input\_if

### data\_input\_input\_tx.sv

**class** input\_tx **extends** uvm\_sequence\_item**;**

`uvm\_object\_utils**(**input\_tx**)**

// To include variables in copy, compare, print, record, pack, unpack, and compare2string, define them using trans\_var in file data\_input.tpl

// To exclude variables from compare, pack, and unpack methods, define them using trans\_meta in file data\_input.tpl

// Transaction variables

**rand** **logic** **[**15**:**0**]** data**;**

**constraint** c\_data **{** 0 **<=** data**;** data **<** 128**;** **}**

**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 data\_input.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 data\_input.tpl

**endclass** **:** input\_tx

**function** input\_tx**::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 data\_input.tpl

**function** **void** input\_tx**::**do\_copy**(**uvm\_object rhs**);**

input\_tx rhs\_**;**

**if** **(!**$cast**(**rhs\_**,** rhs**))**

`uvm\_fatal**(**get\_type\_name**(),** "Cast of rhs object failed"**)**

**super.**do\_copy**(**rhs**);**

data **=** rhs\_**.**data**;**

**endfunction** **:** do\_copy

**function** **bit** input\_tx**::**do\_compare**(**uvm\_object rhs**,** uvm\_comparer comparer**);**

**bit** result**;**

input\_tx 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**(**"data"**,** data**,** rhs\_**.**data**,** $bits**(**data**));**

**return** result**;**

**endfunction** **:** do\_compare

**function** **void** input\_tx**::**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** input\_tx**::**do\_record**(**uvm\_recorder recorder**);**

**super.**do\_record**(**recorder**);**

// Use the record macros to record the item fields:

`uvm\_record\_field**(**"data"**,** data**)**

**endfunction** **:** do\_record

**function** **void** input\_tx**::**do\_pack**(**uvm\_packer packer**);**

**super.**do\_pack**(**packer**);**

`uvm\_pack\_int**(**data**)**

**endfunction** **:** do\_pack

**function** **void** input\_tx**::**do\_unpack**(**uvm\_packer packer**);**

**super.**do\_unpack**(**packer**);**

`uvm\_unpack\_int**(**data**)**

**endfunction** **:** do\_unpack

**function** **string** input\_tx**::**convert2string**();**

**string** s**;**

$sformat**(**s**,** "%s\n"**,** **super.**convert2string**());**

$sformat**(**s**,** **{**"%s\n"**,**

"data = 'h%0h 'd%0d\n"**},**

get\_full\_name**(),** data**,** data**);**

**return** s**;**

**endfunction** **:** convert2string

### data\_input\_monitor.sv

**class** data\_input\_monitor **extends** uvm\_monitor**;**

`uvm\_component\_utils**(**data\_input\_monitor**)**

**virtual** data\_input\_if vif**;**

uvm\_analysis\_port **#(**input\_tx**)** analysis\_port**;**

input\_tx 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 data\_input.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 data\_input.tpl

**endclass** **:** data\_input\_monitor

**function** data\_input\_monitor**::new(string** name**,** uvm\_component parent**);**

**super.**new**(**name**,** parent**);**

analysis\_port **=** **new(**"analysis\_port"**,** **this);**

**endfunction** **:** **new**

**function** **void** data\_input\_monitor**::**build\_phase**(**uvm\_phase phase**);**

**endfunction** **:** build\_phase

**task** data\_input\_monitor**::**run\_phase**(**uvm\_phase phase**);**

`uvm\_info**(**get\_type\_name**(),** "run\_phase"**,** UVM\_HIGH**)**

m\_trans **=** input\_tx**::**type\_id**::**create**(**"m\_trans"**);**

do\_mon**();**

**endtask** **:** run\_phase

// Start of inlined include file generated\_tb/tb/include/data\_input\_do\_mon.sv

**task** data\_input\_monitor**::**do\_mon**;**

**forever** **@(posedge** vif**.**clk**)**

**begin**

**wait** **(**vif**.**reset **==** 1**);**

**if** **(**vif**.**valid **&&** vif**.**ready**)**

**begin**

m\_trans**.**data **=** vif**.**data**;**

analysis\_port**.**write**(**m\_trans**);**

`uvm\_info**(**get\_type\_name**(),** $sformatf**(**"Input data = %0d"**,** m\_trans**.**data**),** UVM\_HIGH**)**

**end**

**end**

**endtask**

### data\_input\_pkg.sv

**package** data\_input\_pkg**;**

`include "uvm\_macros.svh"

**import** uvm\_pkg**::\*;**

`include "data\_input\_input\_tx.sv"

`include "data\_input\_config.sv"

`include "data\_input\_driver.sv"

`include "data\_input\_monitor.sv"

`include "data\_input\_sequencer.sv"

`include "data\_input\_coverage.sv"

`include "data\_input\_agent.sv"

`include "data\_input\_seq\_lib.sv"

**endpackage** **:** data\_input\_pkg

### data\_input\_seq\_lib.sv

**class** data\_input\_default\_seq **extends** uvm\_sequence **#(**input\_tx**);**

`uvm\_object\_utils**(**data\_input\_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** **:** data\_input\_default\_seq

**function** data\_input\_default\_seq**::new(string** name **=** ""**);**

**super.**new**(**name**);**

**endfunction** **:** **new**

**task** data\_input\_default\_seq**::**body**();**

`uvm\_info**(**get\_type\_name**(),** "Default sequence starting"**,** UVM\_HIGH**)**

req **=** input\_tx**::**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 data\_input\_default\_seq**::**get\_starting\_phase**();**

**return** starting\_phase**;**

**endfunction:** get\_starting\_phase

**function** **void** data\_input\_default\_seq**::**set\_starting\_phase**(**uvm\_phase phase**);**

starting\_phase **=** phase**;**

**endfunction:** set\_starting\_phase

### data\_input\_sequencer.sv

// Sequencer class is specialization of uvm\_sequencer

**typedef** uvm\_sequencer **#(**input\_tx**)** data\_input\_sequencer\_t**;**

data\_output

### data\_output\_agent.sv

**class** data\_output\_agent **extends** uvm\_agent**;**

`uvm\_component\_utils**(**data\_output\_agent**)**

uvm\_analysis\_port **#(**output\_tx**)** analysis\_port**;**

data\_output\_config m\_config**;**

data\_output\_sequencer\_t m\_sequencer**;**

data\_output\_driver m\_driver**;**

data\_output\_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 data\_output.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 data\_output.tpl

**endclass** **:** data\_output\_agent

**function** data\_output\_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 data\_output.tpl

**function** **void** data\_output\_agent**::**build\_phase**(**uvm\_phase phase**);**

// You can insert code here by setting agent\_prepend\_to\_build\_phase in file data\_output.tpl

**if** **(!**uvm\_config\_db **#(**data\_output\_config**)::**get**(this,** ""**,** "config"**,** m\_config**))**

`uvm\_error**(**get\_type\_name**(),** "data\_output config not found"**)**

m\_monitor **=** data\_output\_monitor **::**type\_id**::**create**(**"m\_monitor"**,** **this);**

**if** **(**get\_is\_active**()** **==** UVM\_ACTIVE**)**

**begin**

m\_driver **=** data\_output\_driver **::**type\_id**::**create**(**"m\_driver"**,** **this);**

m\_sequencer **=** data\_output\_sequencer\_t**::**type\_id**::**create**(**"m\_sequencer"**,** **this);**

**end**

// You can insert code here by setting agent\_append\_to\_build\_phase in file data\_output.tpl

**endfunction** **:** build\_phase

**function** **void** data\_output\_agent**::**connect\_phase**(**uvm\_phase phase**);**

**if** **(**m\_config**.**vif **==** **null)**

`uvm\_warning**(**get\_type\_name**(),** "data\_output 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 data\_output.tpl

**endfunction** **:** connect\_phase

**function** uvm\_active\_passive\_enum data\_output\_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

### data\_output\_config.sv

**class** data\_output\_config **extends** uvm\_object**;**

// Do not register config class with the factory

**virtual** data\_output\_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 data\_output.tpl

// You can remove new by setting agent\_config\_generate\_methods\_inside\_class = no in file data\_output.tpl

**extern** **function** **new(string** name **=** ""**);**

// You can insert code here by setting agent\_config\_inc\_inside\_class in file data\_output.tpl

**endclass** **:** data\_output\_config

// You can remove new by setting agent\_config\_generate\_methods\_after\_class = no in file data\_output.tpl

**function** data\_output\_config**::new(string** name **=** ""**);**

**super.**new**(**name**);**

**endfunction** **:** **new**

### data\_output\_coverage.sv

**class** data\_output\_coverage **extends** uvm\_subscriber **#(**output\_tx**);**

`uvm\_component\_utils**(**data\_output\_coverage**)**

data\_output\_config m\_config**;**

**bit** m\_is\_covered**;**

output\_tx m\_item**;**

// You can replace covergroup m\_cov by setting agent\_cover\_inc in file data\_output.tpl

// or remove covergroup m\_cov by setting agent\_cover\_generate\_methods\_inside\_class = no in file data\_output.tpl

**covergroup** m\_cov**;**

option**.**per\_instance **=** 1**;**

// You may insert additional coverpoints here ...

cp\_data**:** **coverpoint** m\_item**.**data**;**

// 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 data\_output.tpl

**extern** **function** **new(string** name**,** uvm\_component parent**);**

**extern** **function** **void** write**(input** output\_tx 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 data\_output.tpl

**endclass** **:** data\_output\_coverage

// You can remove new, write, and report\_phase by setting agent\_cover\_generate\_methods\_after\_class = no in file data\_output.tpl

**function** data\_output\_coverage**::new(string** name**,** uvm\_component parent**);**

**super.**new**(**name**,** parent**);**

m\_is\_covered **=** 0**;**

m\_cov **=** **new();**

**endfunction** **:** **new**

**function** **void** data\_output\_coverage**::**write**(input** output\_tx 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** data\_output\_coverage**::**build\_phase**(**uvm\_phase phase**);**

**if** **(!**uvm\_config\_db **#(**data\_output\_config**)::**get**(this,** ""**,** "config"**,** m\_config**))**

`uvm\_error**(**get\_type\_name**(),** "data\_output config not found"**)**

**endfunction** **:** build\_phase

**function** **void** data\_output\_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

### data\_output\_driver.sv

**class** data\_output\_driver **extends** uvm\_driver **#(**output\_tx**);**

`uvm\_component\_utils**(**data\_output\_driver**)**

**virtual** data\_output\_if vif**;**

**extern** **function** **new(string** name**,** uvm\_component parent**);**

// Start of inlined include file generated\_tb/tb/include/data\_output\_driver\_inc\_inside\_class.sv

**extern** **task** run\_phase**(**uvm\_phase phase**);**

// End of inlined include file

**endclass** **:** data\_output\_driver

**function** data\_output\_driver**::new(string** name**,** uvm\_component parent**);**

**super.**new**(**name**,** parent**);**

**endfunction** **:** **new**

// Start of inlined include file generated\_tb/tb/include/data\_output\_driver\_inc\_after\_class.sv

**task** data\_output\_driver**::**run\_phase**(**uvm\_phase phase**);**

`uvm\_info**(**get\_type\_name**(),** "run\_phase"**,** UVM\_HIGH**)**

**forever** **@(posedge** vif**.**clk**)**

**begin**

seq\_item\_port**.**get\_next\_item**(**req**);**

phase**.**raise\_objection**(this);**

vif**.**ready **<=** 1**;**

**wait** **(**vif**.**reset **==** 1**);**

**fork**

**begin**

**repeat** **(**10**)** **@(posedge** vif**.**clk**);**

phase**.**drop\_objection**(this);**

**end**

**join\_none**

seq\_item\_port**.**item\_done**();**

**end**

**endtask** **:** run\_phase

### data\_output\_if.sv

**interface** data\_output\_if**();**

**timeunit** 1ns**;**

**timeprecision** 1ps**;**

**import** data\_output\_pkg**::\*;**

**logic** last**;**

**logic** valid**;**

**logic** ready**;**

**logic** **[**15**:**0**]** data**;**

**logic** clk**;**

**logic** reset**;**

// You can insert properties and assertions here

// You can insert code here by setting if\_inc\_inside\_interface in file data\_output.tpl

**endinterface** **:** data\_output\_if

### data\_output\_monitor.sv

**class** data\_output\_monitor **extends** uvm\_monitor**;**

`uvm\_component\_utils**(**data\_output\_monitor**)**

**virtual** data\_output\_if vif**;**

uvm\_analysis\_port **#(**output\_tx**)** analysis\_port**;**

output\_tx 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 data\_output.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 data\_output.tpl

**endclass** **:** data\_output\_monitor

**function** data\_output\_monitor**::new(string** name**,** uvm\_component parent**);**

**super.**new**(**name**,** parent**);**

analysis\_port **=** **new(**"analysis\_port"**,** **this);**

**endfunction** **:** **new**

**function** **void** data\_output\_monitor**::**build\_phase**(**uvm\_phase phase**);**

**endfunction** **:** build\_phase

**task** data\_output\_monitor**::**run\_phase**(**uvm\_phase phase**);**

`uvm\_info**(**get\_type\_name**(),** "run\_phase"**,** UVM\_HIGH**)**

m\_trans **=** output\_tx**::**type\_id**::**create**(**"m\_trans"**);**

do\_mon**();**

**endtask** **:** run\_phase

// Start of inlined include file generated\_tb/tb/include/data\_output\_do\_mon.sv

**task** data\_output\_monitor**::**do\_mon**;**

**forever** **@(posedge** vif**.**clk**)**

**begin**

**wait** **(**vif**.**reset **==** 1**);**

**if** **(**vif**.**valid **&&** vif**.**ready**)**

**begin**

m\_trans**.**data **=** vif**.**data**;**

analysis\_port**.**write**(**m\_trans**);**

`uvm\_info**(**get\_type\_name**(),** $sformatf**(**"Output data = %0d"**,**m\_trans**.**data**),** UVM\_HIGH**)**

**end**

**end**

**endtask**

### data\_output\_output\_tx.sv

**class** output\_tx **extends** uvm\_sequence\_item**;**

`uvm\_object\_utils**(**output\_tx**)**

// To include variables in copy, compare, print, record, pack, unpack, and compare2string, define them using trans\_var in file data\_output.tpl

// To exclude variables from compare, pack, and unpack methods, define them using trans\_meta in file data\_output.tpl

// Transaction variables

**rand** **logic** **[**15**:**0**]** data**;**

**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 data\_output.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 data\_output.tpl

**endclass** **:** output\_tx

**function** output\_tx**::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 data\_output.tpl

**function** **void** output\_tx**::**do\_copy**(**uvm\_object rhs**);**

output\_tx rhs\_**;**

**if** **(!**$cast**(**rhs\_**,** rhs**))**

`uvm\_fatal**(**get\_type\_name**(),** "Cast of rhs object failed"**)**

**super.**do\_copy**(**rhs**);**

data **=** rhs\_**.**data**;**

**endfunction** **:** do\_copy

**function** **bit** output\_tx**::**do\_compare**(**uvm\_object rhs**,** uvm\_comparer comparer**);**

**bit** result**;**

output\_tx 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**(**"data"**,** data**,** rhs\_**.**data**,** $bits**(**data**));**

**return** result**;**

**endfunction** **:** do\_compare

**function** **void** output\_tx**::**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** output\_tx**::**do\_record**(**uvm\_recorder recorder**);**

**super.**do\_record**(**recorder**);**

// Use the record macros to record the item fields:

`uvm\_record\_field**(**"data"**,** data**)**

**endfunction** **:** do\_record

**function** **void** output\_tx**::**do\_pack**(**uvm\_packer packer**);**

**super.**do\_pack**(**packer**);**

`uvm\_pack\_int**(**data**)**

**endfunction** **:** do\_pack

**function** **void** output\_tx**::**do\_unpack**(**uvm\_packer packer**);**

**super.**do\_unpack**(**packer**);**

`uvm\_unpack\_int**(**data**)**

**endfunction** **:** do\_unpack

**function** **string** output\_tx**::**convert2string**();**

**string** s**;**

$sformat**(**s**,** "%s\n"**,** **super.**convert2string**());**

$sformat**(**s**,** **{**"%s\n"**,**

"data = 'h%0h 'd%0d\n"**},**

get\_full\_name**(),** data**,** data**);**

**return** s**;**

**endfunction** **:** convert2string

### data\_output\_pkg.sv

**package** data\_output\_pkg**;**

`include "uvm\_macros.svh"

**import** uvm\_pkg**::\*;**

`include "data\_output\_output\_tx.sv"

`include "data\_output\_config.sv"

`include "data\_output\_driver.sv"

`include "data\_output\_monitor.sv"

`include "data\_output\_sequencer.sv"

`include "data\_output\_coverage.sv"

`include "data\_output\_agent.sv"

`include "data\_output\_seq\_lib.sv"

**endpackage** **:** data\_output\_pkg

### data\_output\_seq\_lib.sv

**class** data\_output\_default\_seq **extends** uvm\_sequence **#(**output\_tx**);**

`uvm\_object\_utils**(**data\_output\_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** **:** data\_output\_default\_seq

**function** data\_output\_default\_seq**::new(string** name **=** ""**);**

**super.**new**(**name**);**

**endfunction** **:** **new**

**task** data\_output\_default\_seq**::**body**();**

`uvm\_info**(**get\_type\_name**(),** "Default sequence starting"**,** UVM\_HIGH**)**

req **=** output\_tx**::**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 data\_output\_default\_seq**::**get\_starting\_phase**();**

**return** starting\_phase**;**

**endfunction:** get\_starting\_phase

**function** **void** data\_output\_default\_seq**::**set\_starting\_phase**(**uvm\_phase phase**);**

starting\_phase **=** phase**;**

**endfunction:** set\_starting\_phase

### data\_output\_sequencer.sv

// Sequencer class is specialization of uvm\_sequencer

**typedef** uvm\_sequencer **#(**output\_tx**)** data\_output\_sequencer\_t**;**

include

### data\_input\_cover\_inc.sv

**covergroup** m\_cov**;**

option**.**per\_instance **=** 1**;**

cp\_data**:** **coverpoint** m\_item**.**data **{**

**bins** data\_values**[]** **=** **{[**0**:**127**]};**

**}**

**endgroup**

### data\_input\_do\_mon.sv

**task** data\_input\_monitor**::**do\_mon**;**

**forever** **@(posedge** vif**.**clk**)**

**begin**

**wait** **(**vif**.**reset **==** 1**);**

**if** **(**vif**.**valid **&&** vif**.**ready**)**

**begin**

m\_trans**.**data **=** vif**.**data**;**

analysis\_port**.**write**(**m\_trans**);**

`uvm\_info**(**get\_type\_name**(),** $sformatf**(**"Input data = %0d"**,** m\_trans**.**data**),** UVM\_HIGH**)**

**end**

**end**

**endtask**

### data\_input\_driver\_inc\_after\_class.sv

**task** data\_input\_driver**::**run\_phase**(**uvm\_phase phase**);**

`uvm\_info**(**get\_type\_name**(),** "run\_phase"**,** UVM\_HIGH**)**

**forever** **@(posedge** vif**.**clk**)**

**begin**

seq\_item\_port**.**get\_next\_item**(**req**);**

phase**.**raise\_objection**(this);**

**wait** **(**vif**.**reset **==** 1**);**

vif**.**data **<=** req**.**data**;**

vif**.**valid **<=** 1**;**

vif**.**last **<=** 0**;**

**wait** **(**vif**.**ready **==** 1**);**

**fork**

**begin**

**repeat** **(**10**)** **@(posedge** vif**.**clk**);**

phase**.**drop\_objection**(this);**

**end**

**join\_none**

seq\_item\_port**.**item\_done**();**

**end**

**endtask** **:** run\_phase

### data\_input\_driver\_inc\_inside\_class.sv

**extern** **task** run\_phase**(**uvm\_phase phase**);**

### data\_output\_do\_mon.sv

**task** data\_output\_monitor**::**do\_mon**;**

**forever** **@(posedge** vif**.**clk**)**

**begin**

**wait** **(**vif**.**reset **==** 1**);**

**if** **(**vif**.**valid **&&** vif**.**ready**)**

**begin**

m\_trans**.**data **=** vif**.**data**;**

analysis\_port**.**write**(**m\_trans**);**

`uvm\_info**(**get\_type\_name**(),** $sformatf**(**"Output data = %0d"**,**m\_trans**.**data**),** UVM\_HIGH**)**

**end**

**end**

**endtask**

### data\_output\_driver\_inc\_after\_class.sv

**task** data\_output\_driver**::**run\_phase**(**uvm\_phase phase**);**

`uvm\_info**(**get\_type\_name**(),** "run\_phase"**,** UVM\_HIGH**)**

**forever** **@(posedge** vif**.**clk**)**

**begin**

seq\_item\_port**.**get\_next\_item**(**req**);**

phase**.**raise\_objection**(this);**

vif**.**ready **<=** 1**;**

**wait** **(**vif**.**reset **==** 1**);**

**fork**

**begin**

**repeat** **(**10**)** **@(posedge** vif**.**clk**);**

phase**.**drop\_objection**(this);**

**end**

**join\_none**

seq\_item\_port**.**item\_done**();**

**end**

**endtask** **:** run\_phase

### data\_output\_driver\_inc\_inside\_class.sv

**extern** **task** run\_phase**(**uvm\_phase phase**);**

### reference\_inc\_after\_class.sv

**function** **void** reference**::**write\_reference\_0**(**input\_tx t**);**

send**(**t**);**

**endfunction**

**function** **void** reference**::**send**(**input\_tx t**);**

output\_tx tx**;**

tx **=** output\_tx**::**type\_id**::**create**(**"tx"**);**

**if** **(**init\_flag **==** 1**)**

**begin**

init\_flag **=** 0**;**

**foreach(**tx\_save**[**j**])**

tx\_save**[**j**]** **=** 0**;**

**end**

**if** **(**save\_pnt **==** 5**)**

save\_pnt **=** 0**;**

**else**

save\_pnt**++;**

tx\_save**[**save\_pnt**]** **=** t**.**data**;**

tx**.**data **=** tx\_save**[**0**]** **+** tx\_save**[**1**]** **+** tx\_save**[**2**]** **+** tx\_save**[**3**]** **+** tx\_save**[**4**]** **+** tx\_save**[**5**];**

analysis\_port\_0**.**write**(**tx**);**

`uvm\_info**(**get\_type\_name**(),** $sformatf**(**"Reference Model save\_pnt = %0d, data = %0d"**,**save\_pnt**,** tx**.**data**),** UVM\_HIGH**)**

**endfunction**

### reference\_inc\_inside\_class.sv

**extern** **function** **void** send**(**input\_tx t**);**

**int** save\_pnt **=** 5**;**

**logic** **[**15**:**0**]** tx\_save **[**0**:**5**];**

**int** init\_flag **=** 1**;**

2. OpenCores LPFFIR project SVN repository: <https://opencores.org/projects/lpffir>
3. Doulos. *Easier UVM*. Retrieved from <https://www.doulos.com/knowhow/sysverilog/uvm/easier/>
4. EDA Playground LPFFIR project UVM simulations <https://www.edaplayground.com/x/4RFv>
5. ARM publications (2010). AMBA 4 AXI4-Stream Protocol Specification Version 1.0 (ARM IHI 0051)