Ripple Carry Adder Easier UVM

Author: Vladimir Armstrong
vladimirarmstrong@opencores.org

Rev. 1.0
March 19, 2019

This page has been intentionally left blank.

Revision History

Rev.	Date	Author	Description
1.0	03/19/19	Vladimir Armstrong	First Draft

Contents

1 Introduction

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

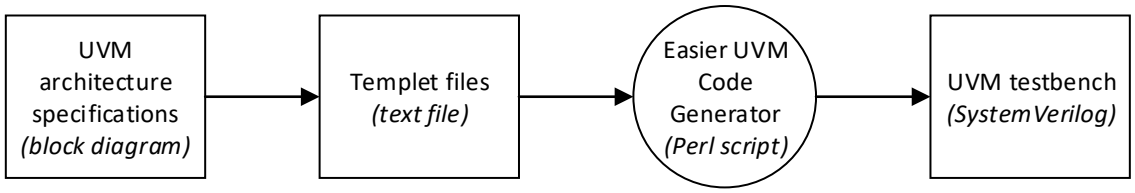

Figure 1 Easier UVM verification flow.

2

UVM Architecture Specifications

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

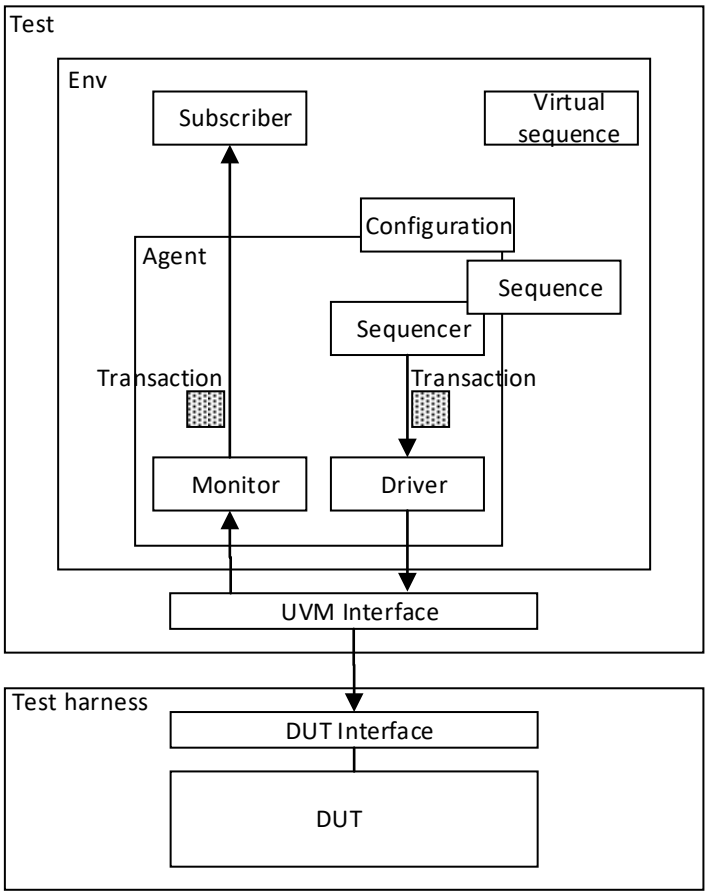

Figure 2 UVM architecture specifications.

3

UVM Code Generator

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

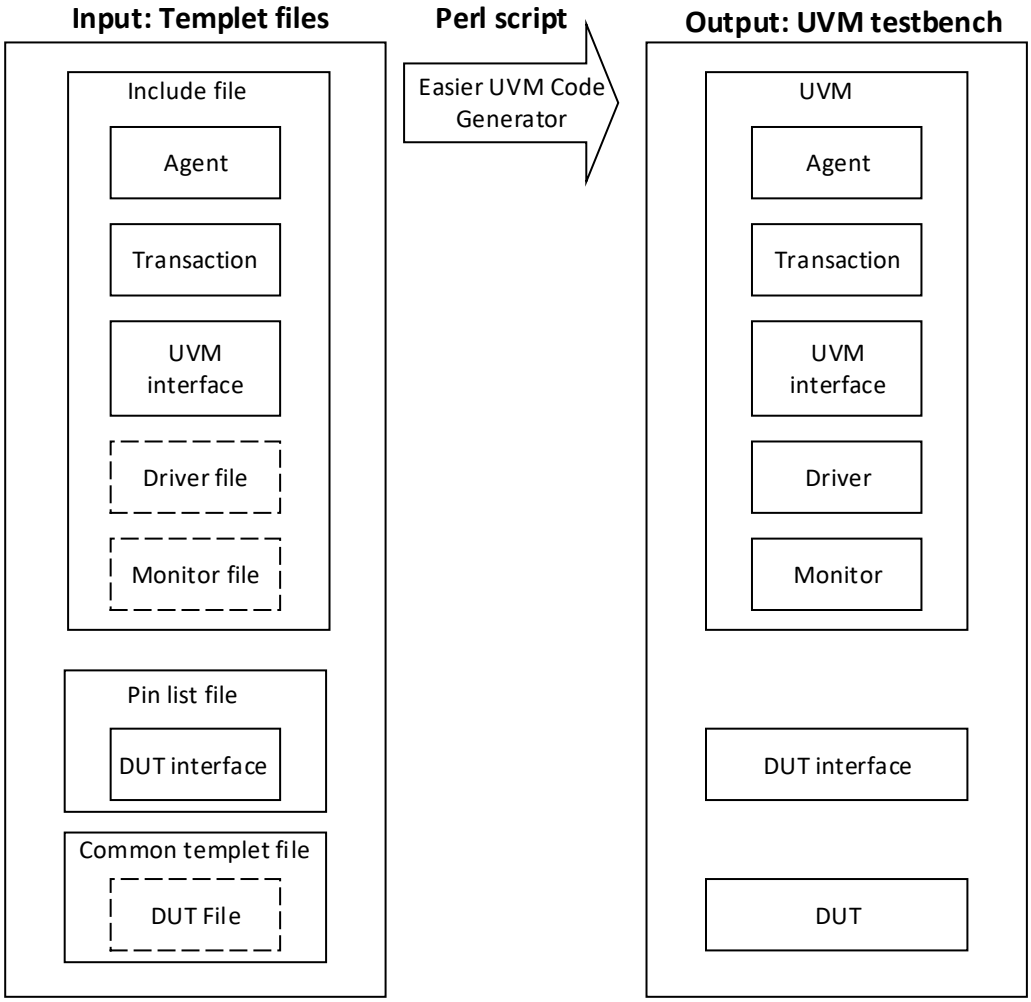

Figure 3 Easier UVM Code Generator.

4

Templet Files

Include File: *rca.tpl*

```
# Agent
agent_name = rca

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

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

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

# Test Clock
if_clock = clk

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

Driver File: *rca_driver_inc.sv*

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

Monitor File: *rca_monitor_inc.sv*

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

Pin List File: *pinlist*

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

Common Templet File: *common.tpl*

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

DUT File: *design.sv*

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

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

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

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

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

endmodule
```

5

UVM Testbench

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

Figure 4 UVM testbench

Appendix A

Simulation Results

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

Index

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