**Testbench for Combinational Adder**

Testbench

////////////////////////// Testbench Code

`timescale 1ns / 1ps

/////////////////////////Transaction

`include "uvm\_macros.svh"

import uvm\_pkg::\*;

class transaction extends uvm\_sequence\_item;

rand bit [3:0] a;

rand bit [3:0] b;

bit [4:0] y;

function new(input string path = "transaction");

[super.new](http://super.new)(path);

endfunction

`uvm\_object\_utils\_begin(transaction)

`uvm\_field\_int(a, UVM\_DEFAULT)

`uvm\_field\_int(b, UVM\_DEFAULT)

`uvm\_field\_int(y, UVM\_DEFAULT)

`uvm\_object\_utils\_end

endclass

//////////////////////////////////////////////////////////////

class generator extends uvm\_sequence #(transaction);

`uvm\_object\_utils(generator)

transaction t;

integer i;

function new(input string path = "generator");

[super.new](http://super.new)(path);

endfunction

virtual task body();

t = transaction::type\_id::create("t");

repeat(10)

   begin

   start\_item(t);

   t.randomize();

   `uvm\_info("GEN",$sformatf("Data send to Driver a :%0d , b :%0d",t.a,t.b), UVM\_NONE);

   finish\_item(t);

   end

endtask

endclass

////////////////////////////////////////////////////////////////////

class driver extends uvm\_driver #(transaction);

`uvm\_component\_utils(driver)

   function new(input string path = "driver", uvm\_component parent = null);

[super.new](http://super.new)(path, parent);

    endfunction

transaction tc;

virtual add\_if aif;

   virtual function void build\_phase(uvm\_phase phase);

     super.build\_phase(phase);

     tc = transaction::type\_id::create("tc");

     if(!uvm\_config\_db #(virtual add\_if)::get(this,"","aif",aif))

     `uvm\_error("DRV","Unable to access uvm\_config\_db");

   endfunction

   virtual task run\_phase(uvm\_phase phase);

   forever begin

   seq\_item\_port.get\_next\_item(tc);

   aif.a <= tc.a;

   aif.b <= tc.b;

     `uvm\_info("DRV", $sformatf("Trigger DUT a: %0d ,b :  %0d",tc.a, tc.b), UVM\_NONE);

   seq\_item\_port.item\_done();

   #10;

   end

   endtask

endclass

////////////////////////////////////////////////////////////////////////

class monitor extends uvm\_monitor;

`uvm\_component\_utils(monitor)

uvm\_analysis\_port #(transaction) send;

function new(input string path = "monitor", uvm\_component parent = null);

[super.new](http://super.new)(path, parent);

   send = new("send", this);

endfunction

transaction t;

virtual add\_if aif;

virtual function void build\_phase(uvm\_phase phase);

  super.build\_phase(phase);

   t = transaction::type\_id::create("t");

  if(!uvm\_config\_db #(virtual add\_if)::get(this,"","aif",aif))

  `uvm\_error("MON","Unable to access uvm\_config\_db");

endfunction

   virtual task run\_phase(uvm\_phase phase);

   forever begin

   #10;

   t.a = aif.a;

   t.b = aif.b;

   t.y = aif.y;

   `uvm\_info("MON", $sformatf("Data send to Scoreboard a : %0d , b : %0d and y : %0d", t.a,t.b,t.y), UVM\_NONE);

   send.write(t);

   end

   endtask

endclass

///////////////////////////////////////////////////////////////////////

class scoreboard extends uvm\_scoreboard;

`uvm\_component\_utils(scoreboard)

uvm\_analysis\_imp #(transaction,scoreboard) recv;

transaction tr;

function new(input string path = "scoreboard", uvm\_component parent = null);

[super.new](http://super.new)(path, parent);

   recv = new("recv", this);

endfunction

virtual function void build\_phase(uvm\_phase phase);

super.build\_phase(phase);

   tr = transaction::type\_id::create("tr");

endfunction

virtual function void write(input transaction t);

  tr = t;

`uvm\_info("SCO",$sformatf("Data rcvd from Monitor a: %0d , b : %0d and y : %0d",tr.a,tr.b,tr.y), UVM\_NONE);

   if(tr.y == tr.a + tr.b)

      `uvm\_info("SCO","Test Passed", UVM\_NONE)

  else

      `uvm\_info("SCO","Test Failed", UVM\_NONE);

  endfunction

endclass

////////////////////////////////////////////////

class agent extends uvm\_agent;

`uvm\_component\_utils(agent)

function new(input string inst = "AGENT", uvm\_component c);

[super.new](http://super.new)(inst, c);

endfunction

monitor m;

driver d;

uvm\_sequencer #(transaction) seqr;

virtual function void build\_phase(uvm\_phase phase);

super.build\_phase(phase);

m = monitor::type\_id::create("m",this);

d = driver::type\_id::create("d",this);

seqr = uvm\_sequencer #(transaction)::type\_id::create("seqr",this);

endfunction

virtual function void connect\_phase(uvm\_phase phase);

super.connect\_phase(phase);

d.seq\_item\_port.connect(seqr.seq\_item\_export);

endfunction

endclass

/////////////////////////////////////////////////////

class env extends uvm\_env;

`uvm\_component\_utils(env)

function new(input string inst = "ENV", uvm\_component c);

[super.new](http://super.new)(inst, c);

endfunction

scoreboard s;

agent a;

virtual function void build\_phase(uvm\_phase phase);

super.build\_phase(phase);

s = scoreboard::type\_id::create("s",this);

a = agent::type\_id::create("a",this);

endfunction

virtual function void connect\_phase(uvm\_phase phase);

super.connect\_phase(phase);

a.m.send.connect(s.recv);

endfunction

endclass

////////////////////////////////////////////

class test extends uvm\_test;

`uvm\_component\_utils(test)

function new(input string inst = "TEST", uvm\_component c);

[super.new](http://super.new)(inst, c);

endfunction

generator gen;

env e;

virtual function void build\_phase(uvm\_phase phase);

super.build\_phase(phase);

gen = generator::type\_id::create("gen");

e = env::type\_id::create("e",this);

endfunction

virtual task run\_phase(uvm\_phase phase);

  phase.raise\_objection(this);

  gen.start(e.a.seqr);

  #50;

  phase.drop\_objection(this);

endtask

endclass

//////////////////////////////////////

module add\_tb();

add\_if aif();

add dut (.a(aif.a), .b(aif.b), .y(aif.y));

initial begin

$dumpfile("dump.vcd");

$dumpvars;

end

initial begin

uvm\_config\_db #(virtual add\_if)::set(null, "uvm\_test\_top.e.a\*", "aif", aif);

run\_test("test");

end

endmodule