SOCGEN Project

Understanding and Using Verilog T*estbenches, Simulations and Toolflows*

Date: Aug 2013

Legal Notices

Tools and documentation are released under the following license

```
#/*****************************************************************/
#/*                                                               */
#/*    Copyright (C) <2013>  <Ouabache Design Works>               */
#/*                                                               */
#/*  This source file may be used and distributed without         */
#/*  restriction provided that this copyright statement is not    */
#/*  removed from the file and that any derivative work contains  */
#/*  the original copyright notice and the associated disclaimer. */
#/*                                                               */
#/*  This source file is free software; you can redistribute it   */
#/*  and/or modify it under the terms of the GNU Lesser General   */
#/*  Public License as published by the Free Software Foundation; */
#/*  either version 2.1 of the License, or (at your option) any   */
#/*  later version.                                               */
#/*                                                               */
#/*  This source is distributed in the hope that it will be       */
#/*  useful, but WITHOUT ANY WARRANTY; without even the implied   */
#/*  warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR      */
#/*  PURPOSE.  See the GNU Lesser General Public License for more */
#/*  details.                                                     */
#/*                                                               */
#/*  You should have received a copy of the GNU Lesser General    */
#/*  Public License along with this source; if not, download it   */
#/*  from http://www.opencores.org/lgpl.shtml                     */
#/*                                                               */
#/*****************************************************************/
```

The opencores.org/Testbench ip are released under the following license.

```
#/*****************************************************************/
#/*                                                               */
#/*                                                               */
#/* Copyright (c) 2013 Ouabache Design Works                      */
#/*                                                               */
#/* All Rights Reserved Worldwide                                 */
#/*                                                               */
#/*Licensed under the Apache License,Version2.0 (the'License');    */
#/*you may not use this file except in compliance with the License.*/
#/*You may obtain a copy of the License at                        */
#/*                                                               */
#/*       http://www.apache.org/licenses/LICENSE-2.0              */
#/*                                                               */
#/*   Unless required by applicable law or agreed to in           */
#/*   writing, software distributed under the License is          */
#/*   distributed on an 'AS IS' BASIS, WITHOUT WARRANTIES         */
#/*   OR CONDITIONS OF ANY KIND, either express or implied.       */
#/*   See the License for the specific language governing         */
#/*   permissions and limitations under the License.              */
#/*****************************************************************/
```

Index

List of Figures

List of Tables

Document versions

Version Number	Date	Author	Summary of Changes
1.00.00	19 AUG 2013	John Eaton	First version.

Tab. 1: Document versions.

Conventions

This document uses the following conventions. An example illustrates each one**.**

Font	Meaning or Usage	Example
`Courier font`	Messages and system displays	`Building sim`
`Courier Bold`	Literal Commands that you enter	
`Courier italic`	Variables where you supply values	

Tab. 2: Conventions.

1 Introduction

A verilog simulator is an essential tool for verifying the proper operation of any digital component. This project provides the tools, design ip and knowledge to create self checking verilog simulations under a socgen development environment.

Toolflows are provided for both the "icarus" and "verilator" simulators and they both use the "covered" code coverage tool to grade the total design coverage of a test suite. A rtl checking toolflow using verilator is also provided.

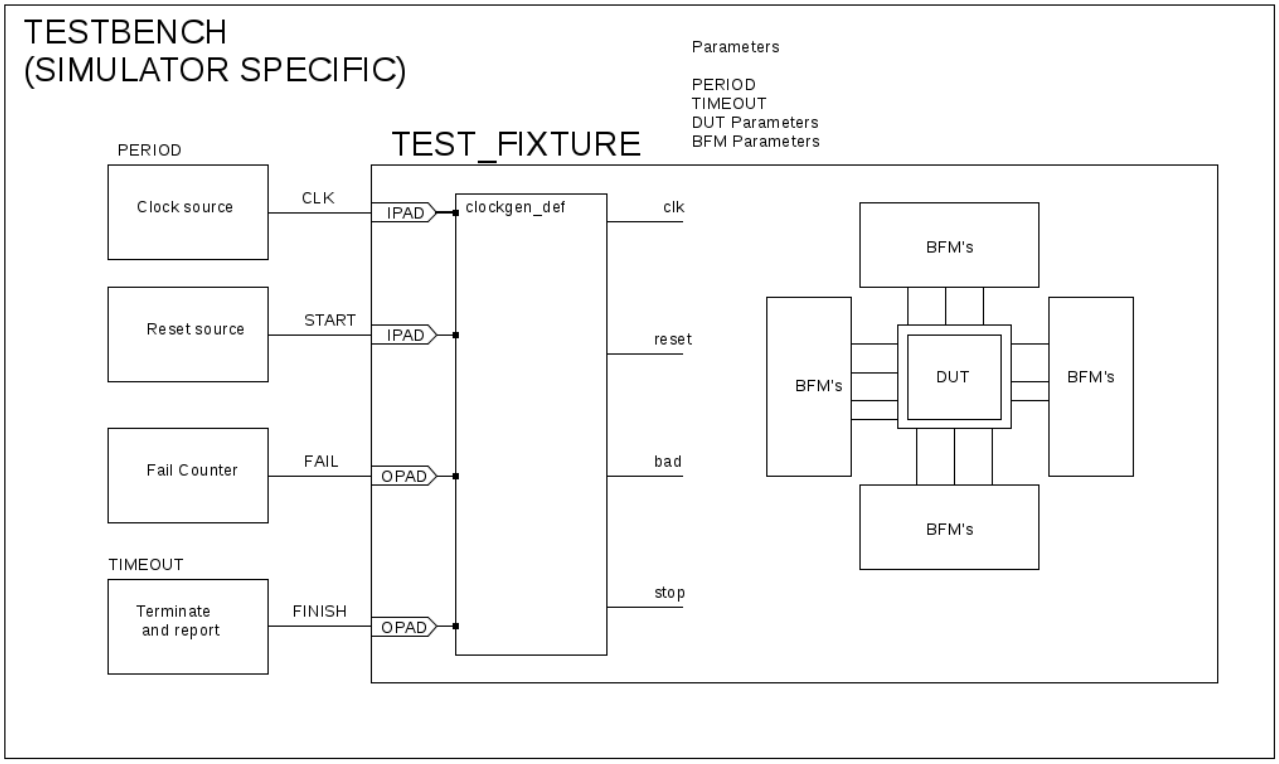
1.1 Contact Information

- Maintainer John Eaton
- Company Ouabache Designworks
- Email z3qmtr45@gmail.com

1.2 Tool Respositories

- Icarus Verilog c
- Verilator c
- Covered c

2 TestBench



2.1 *Creating a Testbench*

A testbench is the top level module in every simulation. Each simulator will have their own specific design requirements so the socgen environment splits it into a top level that contains all simulator specific constructs and a test fixture that can be used on multiple simulators.

There are four signals that connect the testbench to the test fixture.

- clk Master clock signal
- START Asserts to start the simulation.
- FAIL Indicates that DUT provided an incorrect response.
- FINISH Indicates that the simulation has finished.

These four signals connect directly to the clock_def bfm module that converts them into the reset, BAD and STOP signals that connect to all the other bfms.

The testbench also provides parameters to the testfixture. The socgen toolflows will create a custom testbench file for each simulation so that any parameter can be customized for any simulation.

- PERIOD Master clock period in `timescale units
- TIMEOUT Terminate sim if no FINISH signal after this clock count.
- DUT_PARAMs A copy of all the DUT parameters and default values.
- BFM_PARAMs Parameters needed to configure all the BFMs.

2.2 Examples

2.2.1 Icarus Verilog

Scans all xml files in /projects and creates a catalog with the location of all ip-xact and socgen design files. This must be run after any IP is added or removed from the ./projects directory.

```
// Testbench for Mos6502 T6502 T6502_def_tb kim_2 kim_2
`define TIMESCALE 1ns/1ns

`define TIMEFORMAT $timeformat(-6, 2, " us", 14);


`timescale `TIMESCALE

module TB();

initial
begin
`TIMEFORMAT
end

`ifdef VCD
initial
 begin
`include "./dmp_define"
 end
`endif

 reg clk,START;
 wire FAIL,FINISH;
 reg  failed;
 reg [31:0] failcount;
```

```
initial
begin
clk=0;
START=0;
@ (posedge clk);
@ (posedge clk);
@ (posedge clk) ;
START = 1;
end


always@(posedge clk)
if(START && FINISH)
begin
   if(failed)
     begin
        $display("%t    SIM  over:  ERROR       %d  failures",$realtime
,failcount );
     end
  else
     begin
     $display("%t  SIM over: PASSED",$realtime  );
     end // else: !if(failed)
$dumpflush;
$finish;
end


initial
   begin
   next(800000);
   $display("%t  Sim over :ERROR    TIMEOUT",$realtime  );
   @ (posedge clk)
   $dumpflush;
   $finish;
   end
```

```
always@(posedge clk or negedge START)
if(!START)
begin
failed <= 1'b0;
failcount <= 32'h0;
end
else
begin
if(FAIL)
begin
failed <= 1'b1;
failcount <= failcount + 32'h00000001;
end
else
begin
failed <= failed;
failcount <= failcount;
end
end


always  #(40/2) clk = !clk;

`include "./test_define"


     T6502_def_tb
   #(  .ROM_WORDS(128),
       .ROM_ADD(7),
       .ROM_FILE("../sw/table_tim1/table_tim1.abs16"),
       .PROG_ROM_FILE("../sw/kim_2/kim_2.abs16"),
       .PROG_ROM_WORDS(2048),
       .PROG_ROM_ADD(11),
       .STARTUP("../sw/vga_startup_screen/vga_startup_screen.abs"),
       .FONT("../sw/vga_font/vga_font.abs"),
       .PERIOD(40),
       .TIMEOUT(800000)) test
 (.clk(clk),.START(START),.FAIL(FAIL),.FINISH(FINISH));




endmodule
```

2.2.2 Verilator

A verilator testbench is much simpler because verilator performs housekeeping tasks in C++ code.

```
//  Testbench  for  Mos6502 T6502   T6502_def_vtb   kim_2    kim_2

module   TB(   input   clk,input   START,output   FINISH,output   FAIL
);

     T6502_def_vtb
   #(  .ROM_WORDS(128),
       .ROM_ADD(7),
       .ROM_FILE("../sw/table_tim1/table_tim1.abs16"),
       .PROG_ROM_FILE("../sw/kim_2/kim_2.abs16"),
       .PROG_ROM_WORDS(2048),
       .PROG_ROM_ADD(11),
       .STARTUP("../sw/vga_startup_screen/vga_startup_screen.abs"),
       .FONT("../sw/vga_font/vga_font.abs"),
       .TIMEOUT(800000)) test
 (.clk(clk),.START(START),.FAIL(FAIL),.FINISH(FINISH));



`include "./test_define"
endmodule
```

Verilator C++ code.

```
#include "VTB.h"
#include "verilated.h"
#include "verilated_vcd_c.h"


int main(int argc, char **argv, char **env) {
  int i;
  int clk;

  int  timeout = 10000;
  int  period  = 50;

  int  timepassed = 0;
  char result[] ="PASSED          ";
  char fail[]   ="FAILED - TIMEOUT";

  Verilated::commandArgs(argc, argv);
  // init top verilog instance
  VTB* top = new VTB;
  // init trace dump
    Verilated::traceEverOn(true);
    VerilatedVcdC* tfp = new VerilatedVcdC;
    top->trace (tfp, 99);
    tfp->open ("TestBench.vcd");
  // initialize simulation inputs
    //    printf("Hello,World %i   \n",argc);

    if(argc == 2)
      {
     timeout = atoi(argv[1]);
      }

    if(argc == 3)
      {
     timeout = atoi(argv[1]);
        period  = atoi(argv[2]);
      }
```

```
     printf("Simulating timeout     %i    period   %i     \n",timeout
,period);

  top->clk = 0;
  top->START = 0;
  // run simulation till the end
  i=0;
  while ( !top->FINISH   )
      {
      top->START = (i > 12);
      // dump variables into VCD file and toggle clock

          {
       clk = 0;
          tfp->dump (period*i+clk);
          top->clk = !top->clk;
          top->eval ();

       clk = period/2;
          tfp->dump (period*i+clk);
          top->clk = !top->clk;
          top->eval ();

          }
    if (Verilated::gotFinish())  exit(0);
    timepassed++;
    i = i++;
    if(timepassed > timeout)
      {
     strcpy(result,fail);
        break;
      }

      }

  printf("Finished   %i  -  %s   \n", timepassed , result);
   tfp->close();
  exit(0);
}
```

Builds a directory image of the top level design in a volatile work area that is symbolicily linked back to the original files in the repository. It also symbolicly links all child components used in the design in a children directory located below the parent design.

Cleaning the workspace is done by removing all files and rerunning workspace.

Example: **$make workspace opencores.org or1k**

or **./tools/sys/workspace /projects/opencores.org/or1k /work**

This will create a new subdirectory and symbolicly link all of the or1k files under `/work/opencores.org__or1k`.

It will then create another subdirectory called `/work/opencores.org__or1k/children` where it will link all of the subcomponents used anywhere under the top level module.

All of the files generated by any tool flow will be put in this workspace.

2.2.3 BUILD_OBJ

Runs assembler or C-compiler on all firmware in project sw directories.

Example: $**make build_obj**

2.2.4 BUILD_SW

Links all sw libraries and creates bit image files.

Example: $**make build_sw**

3 SOCGEN Meta-data files

SOCGEN uses seven xml meta-data files to hold all the the design and configuration data for all of the IP and these will act as an electronic “Data Sheet” for all the tools. Two of these are custom schema but the remaining five generally follow the IEEE-1685-2009 IP-Xact standard.

3.1 SOCGEN:componentConfiguration file

Each component must have a socgen:componentConfiguration file to provide setup and configuration information for tool flows.

3.1.1 Header

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
//                                      //
// Copyright boilerplate                //
//                                       //
-->
<socgen:componentConfiguration    xmlns:socgen="http://opencores.org"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<socgen:vendor>opencores.org</socgen:vendor>
<socgen:library>fpgas</socgen:library>
<socgen:component>Nexys2_T6502</socgen:component>
```

3.1.2 Configuration sets

Each configuration consists of set of name/value pairs for all of the modifiable parameters. Each component must have one configuration named “default” with all the parameters and their default values. Other configurations may be created as needed for any or all of the tool flows.

```
<socgen:configurations>
<socgen:configuration>
<socgen:name>default</socgen:name>
<socgen:parameters>
<socgen:parameter>
<socgen:name>RAM_WORDS</socgen:name><socgen:value>2048</socgen:value>
</socgen:parameter>
<socgen:parameter>
<socgen:name>RAM_ADD</socgen:name><socgen:value>11</socgen:value>
</socgen:parameter>
</socgen:parameters>
</socgen:configuration>
</socgen:configurations>
```

3.1.3 Sim

Each component must have at least one testbench for use in simulations and linting. The testbench it self will be a separate ip-xact component but this section identifies it as a testbench and provides setup data for code coverage. Code coverage is only preformed on the blocks identified in a cover block.

```
<socgen:sim>
 <socgen:library_path>/ip/Nexys2_T6502/sim</socgen:library_path>
<socgen:testbenches> </socgen:testbenches>
<socgen:lints> </socgen:lints>
<socgen:icarus> </socgen:icarus>
</socgen:sim>
```

3.1.4 Testbenches

Each component must have at least one testbench for use in simulations and linting. The testbench it self will be a separate ip-xact component but this section identifies it as a testbench and provides setup data for code coverage. Code coverage is only preformed on the blocks identified in a cover block.

```
<socgen:testbenches>
<socgen:testbench>
<socgen:variant>Nexys2_T6502_tb</socgen:variant>
<socgen:version>tb</socgen:version>
```

```
<socgen:code_coverage>
  <socgen:cover>
  <socgen:name>T6502_cpu_def</socgen:name>
  <socgen:componentInstance>TB.test.dut.core.T6502.cpu<
  /socgen:componentInstance>
  </socgen:cover>
  <socgen:cover>
</socgen:code_coverage>
</socgen:testbench>
</socgen:testbenches>
```

3.1.5 Rtl Checks

This section lists each simulation along with its testbench, configuration and any additional parameters.

```
<socgen:rtl_checks>
      <socgen:lint>
        <socgen:name>io_irq_2</socgen:name>
        <socgen:configuration>T6502_io_irq_2</socgen:configuration>
        <socgen:variant>Nexys2_T6502_lint</socgen:variant>
        <socgen:parameters>
          <socgen:parameter>
              <socgen:name>Param_name</socgen:name>
              <socgen:value>Param_value</socgen:value>
          </socgen:parameter>
        </socgen:parameters>
      </socgen:line>
</socgen:rtl_checks>
```

3.1.6 Simulation Tests

This section lists each simulation along with its testbench, configuration and any additional parameters.

```
<socgen:icarus>
      <socgen:test>
        <socgen:name>io_irq_2</socgen:name>
        <socgen:configuration>T6502_io_irq_2</socgen:configuration>
        <socgen:variant>Nexys2_T6502_tb</socgen:variant>
        <socgen:parameters>
          <socgen:parameter>
              <socgen:name>TIMEOUT</socgen:name>
              <socgen:value>800000</socgen:value>
          </socgen:parameter>
        </socgen:parameters>
      </socgen:test>
</socgen:icarus>
```

3.1.7 syn

This section lists each synthesizable design along with its target technology and overlay library. If the design uses any component listed in the overlay library that also exists in the target library then the target libraries component will be substituted for the original.

```
<socgen:syn>
<socgen:library_path>/ip/Nexys2_T6502/syn</socgen:library_path>

<socgen:ise>
  <socgen:name>Nexys2_T6502_io_irq_2</socgen:name>
  <socgen:target>
    <socgen:vendor>digilentinc.org</socgen:vendor>
    <socgen:library>Nexys2</socgen:library>
    <socgen:component>fpga</socgen:component>
    <socgen:part>xc3s1200e-fg320-5</socgen:part>
    <socgen:overlay>
      <socgen:vendor>opencores.org</socgen:vendor>
      <socgen:library>cde</socgen:library>
    </socgen:overlay>
    </socgen:target>
  </socgen:ise>
</socgen:syn>
```

3.2 IP-Xact spirit:component file

Some of the blocks will be parametrized. This section contains a list of those parameters and default values.

4 References

SOCGEN project
http://opencores.org/project,socgen

IP-Xact standard
http://standards.ieee.org/getieee/1685/download/1685-2009.pdf

Accellera
http://www.accellera.org/apps/org/workgroup/ipxug/download.php/11213/Accellera_Standardized_Bus_Definition_Creation_Workflow.pdf

Accellera
http://www.accellera.org/apps/org/workgroup/ipxug/download.php/11211/Accellera_IP-XACT_Bus_Definition_Creation_Guidelines.pdf5/download/1685-2009.pdf