S3Cbench v.1.0 User Manual (Security Synthesizable SystemC benchmark suite)

Version	Date	Prepared by	Comments
1.0	17/8/2016	Benjamin Carrion Schafer and Nandeesh Veeranna	Initial document

Contents

Getting Started

1.1Introduction

This manual demonstrates the effect of hardware Trojan inserted in behavioral Intellectual Properties (IPs) released by the Design Automation and Reconfigurable Computing Laboratory (DARClab). The benchmarks can be downloaded from sourceforge.net at **http://sourceforge.net/projects/** and are Trojan-infected version of the original S2CBench benchmarks, which can be found at **http://sourceforge.net/projects/s3cbench**. The S3Cbench benchmarks have been designed to show that even 100% code-coverage cannot guarantee the malicious free design**.**

The main idea of these benchmarks is to allow anyone interested in Hardware security and in particular at High-Level Synthesis to have quick access to a larger number of designs with different types of hardware Trojan.

2.1Document Scope

This document is the user manual for the S3CBench benchmark suite. It shows how to install and use the benchmarks.

3.1Licensing

Department of Electrical Engineering
University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 78025
schaferb@ieee.org

error-free. The end-user understands that the program was developed for research purposes and is advised not to rely exclusively on the program for any reason.

IN NO EVENT SHALL THE UNIVERSITY OF TEXAS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN AD- VISED OF THE POSSIBILITY OF SUCH DAMAGE. THE UNIVERSITY OF TEXAS SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN “AS IS” BASIS, AND THE UNIVERSITY OF TEXAS HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

4.1Release Notes

Version 1.0

- S3CBench v.1.0 is the first general release. Please let us know if you encounter any bugs.
- All benchmarks are synthesizable. The HLS tool to synthesize the SystemC designs was CyberWorkBechch from NEC.

5.1Errata

While we have extensively tested the current distribution, we are human and cannot eliminate all bugs in our distribution. As a general rule of thumb, if you find yourself delving into the code, you have gone too far. Contact us if you need additional assistance.

S3CBench Benchmark Overview

6.1S2CBench and S3CBench

In 2013, the DARClab released the S2CBench benchmark suite. The first release (v.1.0) contained 13 benchmarks of different domains written in synthesizable SystemC. Each benchmark was included to test different features of commercial High-Level Synthesis tools, as all of them support SystemC, which makes SystemC the only portable language across all HLS tools.

Figure 1 shows the typical structure of the each of the S2CBench benchmarks. Each benchmarks contains 3 main threads. The first one sends data to the Unit Under Test (UUT), the UUT test itself and a last thread, reads the results returned by the UUT. Before finishing the simulation the testbench compares the results with a golden output.

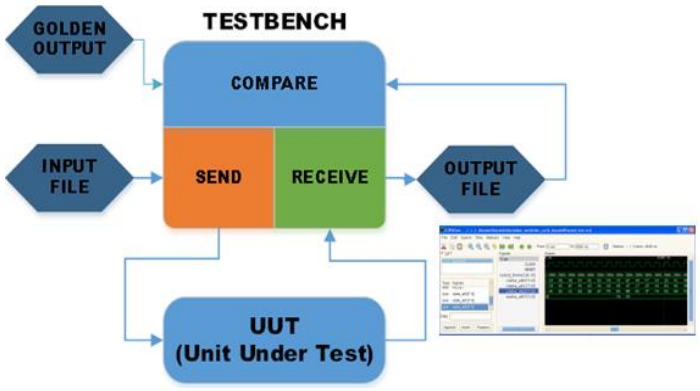


Figure 1 S2CBench benchmark structure

S3C bench is a Hardware Trojan-infected version of S2Cbcench, which demonstrates the effect of Hardware Trojans inserted in behavioral IPs. These Hardware Trojans are hard to detect during the normal verification process. In the area of hardware security, one can come across different types of following possible Hardware Trojans based on ***Trigger*** and ***payload***.

1. **Combinational Trojan Without Memory (CWOM)**: In this type of Trojan, the trigger mechanism is combinational, i.e. , the Trojan triggers for a particular combination of input and the payload has no memory.
2. **Combinational Trojan With Memory (CWM)**: In this type of Trojan, the trigger mechanism is combinational and the payload has memory.
3. **Sequential Trojan With Memory (SWM)**: In this type of Trojan, the trigger mechanism is sequential, i.e. , the effect of Trojan is going to sustain forever once it has been Triggered based

on the counter reaching certain specified value. Also, the payload has memory. It is also called as ***time bomb*** Trojan.

4. **Sequential Trojan Without Memory (SWOM)**: In this type of Trojan the trigger mechanism is sequential and the payload has no memory. In this type, the Trojan triggers based on particular count value and the effect will not sustain forever like SWM.

The above different types of Trojan have been inserted to 8 different S2C benchmarks. The summary of the Trojan along with the compiler directive options are sown in the table 1 below:

Table 1: S3Cbench and their compiler directive options

Benchmarks	**Script file**	**-m <makefile>**	**-t <Trojan type>**	**-bmp <image file(.bmp)>**	**-sc <SystemC file(.cpp)>**
ADPCM	script_adpcm.pl	Makefile	swom swm	N/A	adpcm_encoder.cpp
AES	script_aes.pl	Makefile	cwom	N/A	aes.cpp
Decimation	script_decim.pl	Makefile	swm	N/A	filt_decim.cpp
Disparity	script_disparity.pl	Makefile	cwom1 cwom2 swm	3D_crysis.bmp stero_2.bmp	disparity.cpp
FIR	script_fir.pl	Makefile	cwom	N/A	fir.cpp
Interpolation	Script_interpolation.pl	Makefile	cwom swom swm	N/A	filter_interp.cpp
Sobel	script_sobel.pl	Makefile	cwom cwm swm	lena512.bmp boat.bmp Einstein.bmp	sobel.cpp
UART	script_uart.pl	Makefile	swm1 swm2	N/A	uart.cpp

NOTE: The compiling option for the Trojan free design is –t none

Downloading S3CBench

7.1Download and Content

The benchmarks can be downloaded from **http://sourceforge.net/projects/s3cbench.**
They are tarred and zipped and can be extracted using and decompression software ore in Linux typing:

```
%tar -xvf  S3CBench_v1_0.tgz
```

The benchmarks suite contains 8 designs. Each folder contains script file(.pl) to automatically execute the flow, makefile, design files(source file and other dependent files(.cpp,.h,.txt)), image files(only for sobel and disparity estimator).

7.2Downloading and Installing SystemC on Linux

1. Download SystemC source from Accellera.
http://accellera.org/downloads/standards/systemc

2. Decompress the package using the command
tar -xvf systemc-2.3.0.tgz

3. Change to the top level directory systemc-2.3.0
$cd systemc-2.3.0

4. Make a directory systemc-2.3.0 for installation in your /usr/local/ path.
$mkdir /usr/local/systemc-2.3.0

5. Make a directory “objdir” in the directory systemc-2.3.0
$mkdir objdir

6. Change to objdir
$cd objdir

7. now type:
export CXX=g++

8. Run configure from objdir:
../configure --prefix=/usr/local/systemc230

9. And then:
make

10. Install by typing command:
sudo make install

7.3Downloading and installing profiling tools

Since this project demonstrates the existence of hardware Trojan even with the 100% (or same as the original design) coverage result, it is necessary to install the following profiling tools to compare the profiling report of original and Trojan free design.

GCOV: GCOV is a source code coverage analysis and statement-by-statement profiling tool. Gcov generates exact counts of the number of times each statement in a program is executed and annotates source code to add instrumentation. Gcov comes as a standard utility with theGNU Compiler Collection (GCC) suite. The installation of gcov is not needed because it comes with GNU gcc compiler.
LCOV: LCOV is a graphical front-end for GCC's coverage testing tool gcov. It collects gcov data for multiple source files and creates HTML pages containing the source code annotated with coverage information. It also adds overview pages for easy navigation within the file structure. LCOV supports statement, function and branch coverage measurement.
Lcov can be downloaded from the following website:
http://ltp.sourceforge.net/coverage/lcov.php

8.1Benchmark execution

Each benchmark contains the following files:

Makefile -- Need to modify the path to the systemC folder
---
make : Generates the executable binary
make wave :Generates the same binary, but the simulate creates a VCD file to view the simulation results
make debug : to create a debug version -- VCD File can be visualied with GTKwave (free VCD file viewer)

make clean : cleans the exe and .o files

SystemC files

---

main.cpp : Instantiates the modules which sends and receives data and the unit under test

benchmark.cpp /.h : Main description of the benchmark

tb_benchmark.cpp /.h : Testbech for the given benchmark

define.h : Includes define statments and stimuli filenames

Stimuli files (.txt)

<name>.txt : File with input stimuli (could be more than one)

<name>_golden.txt : File with golden output with which the simulation results will be compared

Script file(.pl)

<name>.pl : File to execute the complete flow in one stretch.

8.2 Command to execute the complete flow from the terminal

The following command executes the complete flow in a single stretch

perl <script_file.pl> -m <makefile> -sc <source file(.cpp)> -t <Trojan type> -bmp<bitmap image file(only for sobel and disparity)>

Example command for sobel:

perl script_sobel.pl –m Makefile –sc sobel.cpp –t cwom –bmp lena512.bmp

Example command for AES:

perl script_aes.pl –m Makefile –sc aes.cpp –t cwom

Since AES design does not have any image file, -bmp option must be excluded from the command.

The options for –m, -sc, -t, -bmp can be found from Table 1.

NOTE: For the Trojan free(original) design specify –t none.

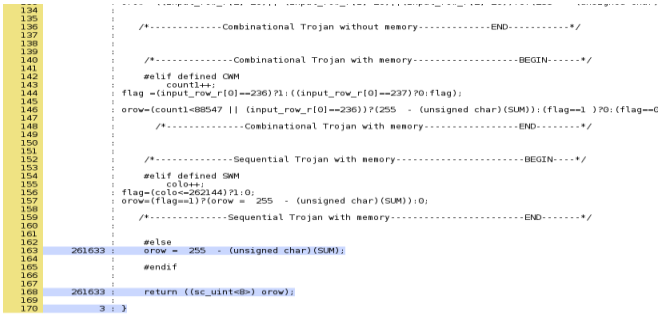
This single command accomplishes the following things:

1. Run the Makefile to compile all the source and dependent files with gcov profiling options.
2. Execute the design(./<source>.exe)

3. Generate the coverage report using the option gcov –r source.cpp
4. Generates the HTML file containing the source code annotated with coverage information using Lcov(Graphical version of Gcov)

Since the IP vendor cannot provide all the possible test cases and their respective golden outputs, the Trojan will not trigger for the inputs provided by the IP vendor. It only triggers for the other inputs outside the golden inputs.

9.1Analyze the Results



```
134            :
135            :
136            :    /*-------------Combinational Trojan without memory------------END----------*/
137            :
138            :
139            :
140            :     /*-------------Combinational Trojan with memory-------------------BEGIN-----*/
141            :
142            :     #elif defined CWM
143            :          count1++;
144            : flag =(input_row_r[0]==236)?1:((input_row_r[0]==237)?0:flag);
145            :
146            : orow=(count1<88547 || (input_row_r[0]==236))?(255  - (unsigned char)(SUM)):(flag==1 )?0:(flag==0
147            :
148            :          /*-------------Combinational Trojan with memory----------------END--------*/
149            :
150            :
151            :
152            :     /*-------------Sequential Trojan with memory----------------------BEGIN----*/
153            :
154            :     #elif defined SWM
155            :          colo++;
156            : flag=(colo<=262144)?1:0;
157            : orow=(flag==1)?(orow =  255  - (unsigned char)(SUM)):0;
158            :
159            :    /*-------------Sequential Trojan with memory----------------------END-------*/
160            :
161            :
162            :     #else
163       261633 :     orow =  255  - (unsigned char)(SUM);
164            :
165            :     #endif
166            :
167            :
168       261633 :     return ((sc_uint<8>) orow);
169            :
170            3 : }
```

LCOV - code coverage report

		Hit	Total	Coverage
Current view: top level - sobel - sobel.cpp (source / functions)	Lines:	36	36	100.0 %
Test: sobel_final.info	Functions:	4	4	100.0 %
Date: 2016-07-29				

```
      Line data    Source code
1              :
2              : //==========================================================================================
3              : //
4              : // File Name    : sobel.cpp
5              : // Description  : Sobel filter implemenetation
6              : // Release Date :
7              : // Author       : PolyU DARC Lab
8              : //                Benjamin Carrion Schafer, Nandeesha V
9              : //
10             : // Revision History
11             : //------------------------------------------------------------------------------------------
12             : // Date        Version    Author           Description
13             : //------------------------------------------------------------------------------------------
14             : //23/07/2013      1.0     PolyU DARC Lab     main sobel definition header
15             : //25/07/2016      2.0     PolyU DARC Lab     Modified for Hardaware Trojan case
16             : //==========================================================================================
17             : #include "define.h"
18             : #include "sobel.h"
19             : int colo;
20             : int count1;
21           5 : void sobel::sobel_main(void)
22             : {
23             :
24          32 :     sc_uint<8> input_row_read[3];
```

[END]