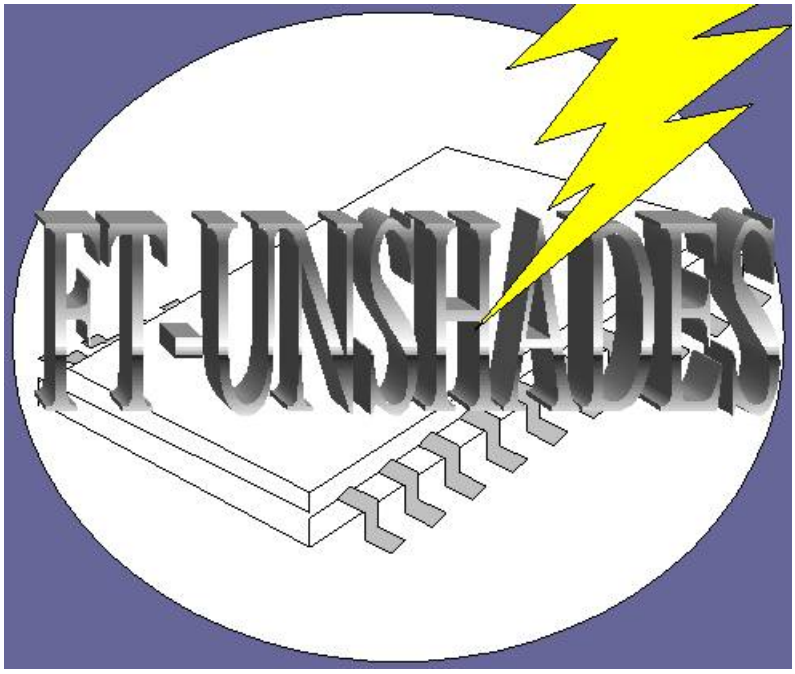
Project Number:	**FT-UNSHADES: ESTEC Contract 17540**
Project Title:	**Fault Tolerant – University of Seville Hardware Debugging System**
Security*:	**Int, Project Members AICIA-GTE, ESA**



Date of Delivery:	2006/02/13
Title:	**CCN2 final document**
Work-Package:	**N/A**
Deliverable Type**:	**DR**
Author(s):	**Jon Tombs, M.A. Aguirre**
	GTE-AICIA
	Camino de los descubrimientos s/n 41095 Sevilla - SPAIN
	tel. +34 95 448 73 76 - fax +34 95 448 73 73
	e-mail: jon@gte.esi.us.es

Abstract:
Results of the CCN2 experience of the FT-UNSHADES project

*Security:	Int.	*Internal circulation within project (and ESA Project Officers if requested)*
	Rest.	*Restricted circulation list (Authorised project reviewers)*
	GTE	*Circulation within GTE-AICIA participants*
	Pub.	*Public document*

**Type: DR-Draft, PR-Prototype, RE-Report, SP-Specification, TO-Tool, OT-Other

FT-UNSHADES

Fault Tolerant Hardware Emulation

“FT-UNSHADES CCN2 Report ”

Final Report
Version 1.2

February 2005

Table Of Contents

Summary Information

Final Report
Version 1.1 / 2 Feb 2006

Classification: Draft
This specification is: Internal.

Contributors to this document:
This document has been prepared by GTE-AICIA, responsible of this deliverable.

History

V1.0	Jan 2006	First Draft, Miguel A. Aguirre & Jon Tombs
V1.1	Feb 2006	Reviews Agustín Fernandez & Francisco Tortosa
V1.2	Feb 2006	Final review and some addings from Miguel A. Aguirre & Jon Tombs

Executive Summary

The FT-UNSHADES project was officially accepted on the 17th April 2005. This document records the progress made within the six months after the acceptance of the project and the experiences over the tested IPs.

1. Introduction

The six months period after the acceptance meeting of the FT-UNSHADES project [1] has been completed, this document records the principal results of each work package currently in progress. The Contract Change Notice number 2 has been signed and delivered in June 2005.

The goal of the test campaign is to insert faults to a complex enough IP in the FT-UNSHADES system, perform a complete test using the capabilities of FT-UNSHADES tools, and proof the emulation approach. The selected IP is Leon2-FT which has been licensed by ESA within the CCN2 scope of FT-UNSHADES. The system has been extensively tested and the results compared with other two similar IPs, Simple Leon2 and Leon2-XTMR, where protections have been inserted to simple Leon2 and by Xilinx using an automatic tool called XTMRtool.

This document presents the raw results for every test and some important conclusions have been obtained. FT-UNSHADES can be a good complement to the real radiation testing, because it can provide important information about the test procedure.

1.1 Scope and Object of this Document

This document is part of the ESA-AICIA agreement for FT-UNSHADES project evaluation.

1.2 References

[1] Fault Tolerant – University of Seville Hardware Debugging System, Project Proposal by AICIA-GTE for ESA-ESTEC.

[2] Fault Tolerant – University of Seville Hardware Debugging System, System requirements Document.

[P1] “Método para análisis de Test Funcional de Circuitos Digitales de Gran Dimensión mediante Emuladores Hardware” International Patent Number: W ES-02/00571

[P2] “Procedimiento para la INDUCCIÓN de valores en los registros de un circuito digital emulado mediante un circuito integrado de EMULACIÓN hardware”. Patent Pending (no. P200102683).

1.3 Legends and Abbreviations

C-FPGA: Configuration FPGA, device design to control all system communications and housekeeping functions. It provides the bridge between the FT-UNSHADES hardware and software.

DTE: Design for Test Emulation. Made out of two instantiations of the design to be tested, and a "wrapper" with control logic.

EPP: Enhanced Parallel Port. Fast, bidirectional mode for the PC parallel port, the use of which is currently in decline.

FF: Flip-Flop, a single edge triggered memory element

FPGA: Field Programmable Gate Array. Reprogrammable array of logic gates and functions that can emulate any digital circuit.

GE: Gold Emulation. It's the theoretical result for the netlist

GR: Gaissler Research

ISP: In System Programming. On board flash memories for self configuration.

MUT: Module Under Test

S-FPGA: System FPGA, large FPGA device, used to emulate the system under test.

SEU: Single Event Upset, the change in a single bit of system state due to the impact of cosmic radiation.

SG: System Gates. Measurement of the netlist size that can be hosted in a FPGA.

SS: System State, the entire state of a system (the value of all of its memory elements in a given moment).

TE: Test Emulation is the emulation of the netlist prepared for being manipulated in the sense of FT-UNSHADES objectives.

TNT: Test and Analysis Tool (software toolbox that operates the FT-UNSHADES system)

USB: Universal Serial Bus. Fast serial bus, currently the most popular link between hardware and PC.

WPX.Y: Task or Input/Output of the current work package.

2. CCN2 Activity

2.1 Project Management

New publications are scheduled reporting the research of fault testing over Leon IP.

Versions 7.X of Xilinx ISE design flow are NOT qualified for FT-UNSHADES purpose. Reason: There are bugs in the SelectMap port management in the bitstream phase. New 8.1, SP2 version scheduled for next January will be the next to be tested, but at the time of this report, the SP2 is not yet available.

Latest version of the TNT (v1.6) software has been generated and sent to ESA.

All tests related in this document have been done using 6.3 version of Xilinx tools and Synplify Pro 8.0

2.2 Project Support: Software Improvements

- Some statistics are inserted for analysis purpose, such as total number of registers, minimum, medium and maximum number of clock cycles that fault need to propagate to primary outputs, and test time.
- New .ll database reading method has been produced. Bi-dimensional busses are now allowed and new internal organisation of the database for speed up the system performance.

2.3 Project Testing over Leon2

- Free distributed version of Leon2 IP downloaded inserted and assembled. An initial functional testing of Leon2 has been performed in order to validate it against the ModelSim simulation. Once the functional test has been passed, fault campaigns have been produced over the complete system and over the first hierarchical level individual blocks of its architecture.
- Test vectors have been recorded for a Modelsim simulation of the Leon2, running with a ROM program called "*full test*", that Gaissler Research provides in the Leon2 package. This is a device production test that exercises all the main functions of Leon2.
- The same core of Leon2 has been sent to Xilinx for being SEU-protected using XTMRtool. As in the case of the simple Leon2, the functional test has been checked against the

simulation of the simple Leon2. The same test campaigns for Leon2-XTMR have been produced. Unfortunately Xilinx didn't sent any information of the conditions and results of the XTMR-tool

- The core for Leon2-FT has been produced with the same settings of the Leon2 (non-FT). Due to GR requirements and the synthesis tool, the internal naming conventions of the post synthesis netlist are substantially different from the previous models.

2.3.1 Test Conditions.

Test vectors have been produced from the testbenches provided by GR. The test programs are the same for the three IPs. Stimuli set have 369848 vectors, where the first 100 vectors are dedicated to reset the complete processor. Since reset is an essential requirement of FT-UNSHADES for all testing models, the "time window" when fault injection takes place has been defined from the vector 100 and vector 369848.

In the case of Leon2-FT the test vector database has been created using the same ROM than Leon2 and Leon2-XTMR. After the corresponding simulation the total number of vectors has grown to be 371843 due to the additional configuration steps needed the FT version of Leon2 (EDAC, parity, etc)

Number of test runs is by default 10000, which is significant enough. The same campaign has been produced for 50000 runs in some experiments, obtaining the same numbers, times 5.

A mixed DTE model, built with simple Leon2 and Leon2-XTMR has been created to test that identical outputs from the two models are obtained if the same test vectors are applied. This model cannot be created with Leon2-FT due to the small differences between the FT and the non-FT structures, and the unavoidable differences in their configuration bitstreams.

<u>The test conditions that are common to all test campaigns are:</u>

1 "test campaign" = 10000 test runs (sometimes, where indicated, more or fewer test runs were executed per test campaign)

1 "test run" = execution of one complete set of test vectors (based on the "full test" test bench provided by GR

Two test campaigns have been executed over the native netlists of Leon2, Leon2-XTMR and Leon2-FT. One with *set SEUSPERRUN=1* condition (single SEU) and a second campaign with *SEUSPERRUN=5*. All faults (emulated SEUs) were injected on a register (or group of registers) which was (were) selected RANDOMLY by FT-UNSHADES amongst all registers present in each targeted module.

The time of the test run when the fault was injected was also chosen randomly by FT-UNSHADES, inside the following time window: from test cycle #100 (after reset) to last test run cycle (369848 or 371843 for Leon2FT)

2.3.2 Leon2 hierarchy

The following table shows the hierarchical distribution of the internal modules. Tests campaigns have been run on all these modules:

leon0/mcore0		
uart1		
uart2		
Proc0	iu0	
	rf	
	c0	dcache
		icache
reset		
Ahb0		
Timers0		
irqctrl0		
apb		
mctrl0		
ioport0		

Some tests has been run over memory modules, but just over the surrounding logic, because the test over BRAM procedure is not yet valid.

Uart1 and Uart2 are treated as a single unit.

2.3.3 Test results summary

Results are reported in terms of number of the test runs (where faults are injected) that provoke any differences at the output signals, the output activity itself (in how many test runs a given output signal was affected by the SEU), mean and maximum number of test cycles (e.g. latency) for a fault to appear at the outputs after being injected. For each test campaign, a list of the number of faults that affected each output is reported.

In the tests of Leon2 (non-FT) many SEUs propagated their effects to many outputs, as indicated in detail in the tables of chapter 2.5.1. In the case of Leon2-XTMR and Leon2-FT none of the emulated SEUs resulted in any differences at any outputs (for test campaigns were only one SEU was injected per test run) . When multiple SEUs were injected in the same test run, some fault cases propagated effects to some of the outputs. This only happened when two or more faults happened to be injected in the same TMR triplet, as expected .

When FT-UNSHADES detects a difference at any of the outputs, the test run stops, and the elapsed time (number of test run cycles) is recorded. Then the next test run with another randomly selected fault starts, and so on, until the 10000 runs are completed and all statistics are obtained. The total time that the group of test runs has taken is given for each test campaing on each module. This time, as can be seen in the following tables is very variable. The reason is that FT-UNSHADES tests speed is very conditioned by the number of programs running on the PC and the CPU activity while the test is running. It also depends heavily on the test vectors themselves, the module which is being evaluated and its likelihood to propagate SEU effects.

Some tests were repeated changing the seed that is used by FT-UNSHADES to do the random selections. This did not seem to have an impact on the statistics obtained.

2.3.4 Test values

Values obtained have been tabulated and presented under the following titles:

Test Conditions: Are the global test conditions to the corresponding module. Number of SEUs per test run, number of test runs per test campaign, clock rates.

Hier. Modules: Represent the subset of registers in the hierarchy modules that are going to be tested. Every column is a different test campaign corresponding to a specific internal module of Leon.

Example of TNT commands for performing the test:

- *TNT> define b=[S*/uart*]*

(this command associates the subset of registers inside the uart module to a bus named ‘b’)

- *TNT> runtest b*

(runtest is run only over the subset of registers associated to ‘b’. SEUs are injected only over this subset)

- *TNT> listn b*

(This command list the complete database of the registers that will be attacked)

Test#: Every test has an assigned code. TXX-Y-Z. XX is the test type, Y is the internal module and Z is the Leon2 version, 0 for simple Leon2, 1 fro XTER version and 2 for FT version. This code will identify every test in the CDROM.

Registers: The total number of registers present in each module involved in the actual test.

Time: Time, in seconds, that lasted the test campaign (all test runs executed sequentially) for each module.

Faults Det: Number of faults that has been detected in the actual test campaign. *Detected* means that during the test run, the injected fault has propagated effects to one or several Leon outputs,

obtaining a discrepancy between the outputs of the GOLD and the SEU-bombarded instantiations of the MUT (Module Under Test).

Mean (cyc): Mean number of system clock cycles (average for all test runs within the same test campaign) which elapsed since a fault was injected until an output mismatch was detected.

Max (cyc): Is the maximum number of system clock cycles that took for a fault to propagate to the outputs.

Name of outputs: The rest of the rows show the number of detected faults in every output, per test campaign (i.e. the addition of all the faults detected after adding up the results from all test runs in the given campaign). The number of lines for every bus is given below:

```
errorn : out std_logic;
address : out std_logic_vector(27 downto 0);
datao : out std_logic_vector(31 downto 0);
dataen : out std_logic_vector(3 downto 0);
ramsn : out std_logic_vector(4 downto 0);
ramoen : out std_logic_vector(4 downto 0);
rwen : out std_logic_vector(3 downto 0);
romsn : out std_logic_vector(1 downto 0);
iosn : out std_logic;
oen : out std_logic;
read : out std_logic;
writen : out std_logic;
pioo : out std_logic_vector(15 downto 0);
pioen : out std_logic_vector(15 downto 0));
```

Reset Mech: This line represents a test that checks if the reset mechanism works properly. There's no global reset in this design, only about 20% of the FFs have asynchronous reset. The principle of the fault injection test is that at the beginning of every test run a system reset should be activated. If there's no global reset an initialisation procedure has to be performed that takes 100 clock cycles. For this reason the faults are injected after the 100th cycle using the command:

- *TNT> set time=[100, 369848]*

If at particular run a fault is detected, then the reset mechanism should erase that fault effect at the beginning of the next run. If not, the fault effect is maintained during the next test and the fault should be repeated. For every fault detected, the reset mechanism has been checked using the global mask command:

- *TNT> set masktoseu=1*

This command means that if the fault persists, it should be detected just when the next fault is injected. This check has been made by "fault dictionary" (log file with all detected faults) direct

visual inspection. This check was done only for multiple SEU injection cases, where only a few faults propagated to the outputs).

2.4 Results

This table shows the synthesis results for each Leon2 IP version (MUT).

	n FF	4LUT	clk rate (est)	BRAMS	MULTIPS	Ios
Leon2	2094		17.664 ns	13	1	169
Leon2Xtmr(*)				39	3	169
Leon2-FT	6225	10537	28.213 ns	14	1	169

- (*)Values for Leon2 XTMR are estimated. Numbers were not reported by Xilinx, but XTMR was applied to the maximum extent.

This table shows the post-place and route numbers when the IP are inserted into the FT-UNSHADES design flow, and converted to the model called DTE.

	n FF	4LUT	Syst Gates	clk rate (ns)	BRAM	MULTIPS
Leon2	5222	15249	2247592	23.748	32	2
Leon2Xtmr	13854	49845	5959292	23.916	84	6
Leon2-FT	13370	22542	2491706	23.820	34	2
Test-Shell (*)	1118	2312		118.765	6	0

- (*)Calculated from the other numbers

2.4.1 Convergence of the number of injections

Previous to any test a simple experience has been produced. The simple Leon2 has been tested with a different number of faults (set MAXRUNS= xxx), time and register randomly selected conditions, different seeds and recording the number of faults, same conditions as T00-0-0. The following table shows the results of the experience:

MAXRUNS	Faults	Percentage
100	24	24%
500	100	20%
1000	187	18.7%
5000	849	16.98%
10000	1704	17.04%
200000	33824	16.912%

Main conclusion is that there's a rapid convergence to a 17% of the faults detected over the unprotected mode of the Leon2. Typically this number will produce a dimension for the test that should give a valid number of amount f faults injected.

2.4.2 Analysis for single injections

The FT-UNSHADES analysis model has been performed in several levels.

1.- Single Level (Set SEUSPERRUN=1.)

Results for Simple Leon2

Test conditions: Simple Leon2, 1 SEU per run, 10000 runs per test.

Test#	T00-0-0	T00-1-0	T00-2-0	T00-3-0	T00-4-0	T00-5-0	T00-6-0	T00-6-0	T00-7-0	T00-8-0	T00-9-0	T00-10-0	T00-11-0
Module	top	ahpb	c0	ic	ioport	iu	mctrl	mctrlb	proc	reset (*)	rf	timers	uarts
#Registers	2279	84	266	166	90	980	131	131	1447	8	64	127	
time (s)	2590.33	2592.44	4109.76	2634.06	2632.17	2950.76	2569.48	3341.82	2516.93	255.96	3424.66	2638.15	2620.16
Faults Det	1704	102	2780	2038	215	1717	3089	3081	1704	809	0	119	1700
Mean (cyc)	26690.8	8.5	1152.8	62.5	178404	26063.1	93762.5	94346.8	14594.0	3	0	171082.3	168239.0
Max (cyc)	362337		224519	12545	360357	325256		368357			0	363594	364879
Errorn	11	0	0	0	0	19	0	0	14	0	0	0	0
Address	566	86	1472	1781	3	762	592	556	839	419	0	99	2
Datao	797	15	1263	149	1	923	1401	1428	833	28	0	20	864
Dataen	37	0	0	0	0	0	629	602	0	8	0	0	0
Ramsn	252	83	415	377	1	322	678	701	329	508	0	98	0
Ramoen	240	83	397	377	1	314	478	500	321	481	0	98	0
Rwen	36	0	0	0	0	0	610	589	0	2	0	0	0
Romsn	15	0	33	24	0	37	5	7	22	6	0	0	0
Iosn	11	0	0	0	0	0	246	230	0	0	0	0	0
Oen	202	83	365	327	1	253	273	256	290	485	0	98	0
Read	9	0	31	1	0	11	77	76	15	14	0	0	0
Writen	26	0	0	0	0	0	492	464	0	2	0	0	0
Pioo	152	1	0	0	211	0	0	0	0	0	0	0	834
Pioen	65	0	0	0	0	0	0	0	0	9	0	0	0

(*) 1000 injections

Note:Mctrl and Mctrlb are the same submodule

For Leon-XTMR

Test conditions: Leon2-XTMR, 1 SEU per run, 1000 runs per test.

Test#	T00-0-1	T00-1-1	T00-2-1	T00-3-1	T00-4-1	T00-5-1	T00-6-1	T00-7-1	T00-8-1	T00-9-1	T00-10-1	T00-11-1
Module	top (10000)	ahpb	c0	ic	Ioport	Iu	mctrl	proc	reset	Rf	timers	Uarts
Registers	6894	252	819	504	270	2946	393	4368	42	192	381	651
Time (s)	3130.84	3056.14	298.19	297.84	297.71	297.02	296.78	309.06	312.97	298.02	295.88	301.28
Faults Det	0	0	0	0	0	0	0	0	0	0	0	0
Mean (cycles)	0	0	0	0	0	0	0	0	0	0	0	0
Max (cycles)	0	0	0	0	0	0	0	0	0	0	0	0

For Leon2-FT

Test conditions: Leon2-FT, 1 SEU per run, 1000 runs per test.

Test#	T00-0-2	T00-1-2	T00-2-2	T00-3-2	T00-4-2	T00-5-2	T00-6-2	T00-7-2	T00-8-2	T00-9-2	T00-10-2	T00-11-2
module	Top (*)	ahpb	c0	ic	ioport	iu	mctrl	proc	reset	Rf	timers	Uarts
registers	6463	39	798	171	363	3093	651	4132	15	pending	381	624
Time (s)	25284.61	174.50	182.76	185.62	209.7	185.86	192.45	248.47	267.09		246.56	248
Faults Det	0	0	0	0	0	0	0	0	0	0	0	0
Mean (cycles)	0	0	0	0	0	0	0	0	0	0	0	0
Max (cycles)	0	0	0	0	0	0	0	0	0	0	0	0

(*) 100000 injections

Note: output activity haven't been reported because is always zero.

2.4.3 Analysis for multiple injections

For Leon2-XTMR (2 SEUs per RUN)

Test conditions: Leon2-XTMR, 2 SEU per run, 1000 runs per test.

Test#	T01-0-1	T01-1-1	T01-2-1	T01-3-1	T01-4-1	T01-5-1	T01-6-1	T01-7-1	T01-8-1	T01-9-1	T01-10-1	T01-11-1
module	top (*)	ahpb	c0	ic	ioport	Iu	mctrl	proc	Reset	rf	timers	uarts
#registers	6894	252	819	504	270	2946	393	4368	42	192	381	651
Time (s)	3130.84	3056.14	298.19	297.84	297.71	297.02	296.78	309.06	312.97	298.02	295.88	301.28
Faults	0	0	0	0	0	0	0	0	0	0	0	0
Mean (cyc)	0	0	0	0	0	0	0	0	0	0	0	0
Max (cycl)	0	0	0	0	0	0	0	0	0	0	0	0

Comment: 2 SEUs per run in poor for a 6894 registers test, because the behaviour is like single SEUs. For this reason we decided to introduce 5 SEUs per run

For Leon2-XTMR (5 SEUs per RUN)

Test conditions: Leon2-XTMR, 5 SEU per run, 1000 runs per test.

Test#	T02-0-1	T02-1-1	T02-2-1	T02-3-1	T02-4-1	T02-5-1	T02-6-1	T02-7-1	T02-8-1	T02-9-1	T02-10-1	T02-11-1
module	top	ahpb	c	icache	ioport	iu	mctrl	proc	reset	rf	timers	uart
Registers	6894(*)	252	819	504	270	2946	393	4368	42	192	381	651
time (s)	4384.27	434.47	504.48	455.69	454.62	447.35	455.79	468.86	446.79	452.16	442.98	445.10
Faults Det	8	0	7	6	1	2	22	1	653	0	0	7
Mean (cycles)	61376	0	3.3	3.8	262854	925.5	148811.2	4	1	0	0	204546.1
Max (cycles)	291146	0	11	7	262854	1835	359581	4	26	0	0	347838
errorn	0	0	0	0	0	0	0	0	6	0	0	0
address	3	0	2	6	0	0	1	1	308	0	0	0
datao	2	0	5	0	0	1	8	0	30	0	0	3
dataen	0	0	0	0	0	0	6	0	3	0	0	0
ramsn	2	0	0	3	0	1	7	1	451	0	0	0

ramoen	2	0	0	3	0	1	5	1	413	0	0	0
rwen	0	0	0	0	0	0	7	0	0	0	0	0
romsn	2	0	0	0	0	1	0	0	8	0	0	0
iosn	0	0	0	0	0	0	1	0	0	0	0	0
Oen	4	0	0	3	0	0	1	1	415	0	0	0
read	0	0	0	0	0	0	0	0	5	0	0	0
writen	0	0	0	0	0	0	3	0	1	0	0	0
pioo	0	0	0	0	1	0	0	0	0	0	0	4
pioen	0	0	0	0	0	0	0	0	8	0	0	0
Reset Mech	OK	Na	OK	OK	OK	OK	OK	OK	OK	Na	Na	OK

(*) 10000 quintuple injections

For Leon2-FT (5 SEUs per Run)

Test#	T02-0-2	T02-1-2	T02-2-2	T02-3-2	T02-4-2	T02-5-2	T02-6-2	T02-7-2	T02-8-2	T02-10-2	T02-11-2
module	Top	ahpb	C	icache	ioport	iu	mctrl	proc	reset	Timer	Uart (*)
Registers	6463	39	798	156	363	3093	651	4132	15	381	627
time (s)	3836.43	377.12	463.41	408.02	409.95	418.43	378.40	413.05	420.02	426.17	3299.72
Faults Det	12	101	5	44	12	0	9	2	612	1	52
Mean (cycles)	78936	830.6	1.4	1.9	1	0	97637.6	15.5	2.5	312465.0	171788.7
Max (cycles)	333974	14637	3	9	1	0	222551	19	7	312465	348004
errorn	0	0	0	0	0	0	0	0	0	0	0
address	3	73	2	41	0	0	3	0	229	1	0
datao	7	15	3	4	0	0	3	2	144	0	22
dataen	0	0	0	0	0	0	2	0	4	0	0
ramsn	2	66	0	12	0	0	2	0	440	1	0
ramoen	2	62	0	11	0	0	1	0	424	1	0
rwen	0	0	0	0	0	0	1	0	3	0	0
romsn	0	0	0	0	0	0	0	0	63	0	0
iosn	0	0	0	0	0	0	0	0	0	0	0
Oen	2	62	0	7	0	0	0	0	441	1	0
read	1	0	0	0	0	0	0	0	20	0	0
writen	0	0	0	0	0	0	1	0	3	0	0
pioo	1	0	0	0	7	0	0	0	0	0	28
pioen	2	0	0	0	5	0	0	0	7	0	0
Restet Mech	OK	OK	OK	OK	OK	Na	OK	OK	OK	OK	OK

(*) 10000 quintuple injections

Comment: reset module seems to be the most critical submodule.

2.4.4 Conclusions on the basic testing

The basic testing shows that the system is effectively protected against single SEUs, it seems not to be weak to any kind of single tests. The Leon2 test shows that a low percentage of faults are really detected by the test, giving a clear idea of the **quality of test.** We'll return to this idea further.

In the five SEUs per run test, many errors have been provoked; all of them are due to the attack to the same cell in different clock domains (more than one FF in the same triplet). The most interesting

conclusion is that **reset and initialisation strategy can erase the error**. Due to the reset cycle programmed in Leon2 the initialisation phase is made using asynchronous reset combined with some instructions. This test also checks if the system can easily recover from an error thanks to this initialisation phase. This seems to be the case indeed.

2.4.5 Convergence of the quality of test

As shown in T00-0-0 experiment only 17% of the injected faults are detected. To have enough confidence of this figure, the same experience has been repeated for different MARUNS. The following table shows the number of faults detected showing the asymptotic convergence to this number.

2.5 Leon2-FT with some protection cells corrupted.

This test consists of the generation of a "damaged" netlist for Leon-FT. A new TMR library has been generated with a non effective voter. In some places the original TMR structure has been substituted by the damaged structure. The "*damage*" is the substitution of the original voter by:

Vote<= D(0) and D(1) or (D2).

This structure represents a non protected fault when D(2) is attacked, but remains untouched in the rest of cases

NOTE: We now call "damage" the effect of the new 'voter' inserted into the netlist.

2.5.1 First Experience

The internal registers whose voting logic was to be damaged were selected randomly, without any logical selection because representing the most common situation during design. The designer doesn't know *a priori* where the faulty logic was located.

This experiment has been very disappointing because no damage was detected after 1000000 test runs faults experience (3-4 days running) under 1 SEU per run condition.

The post synthesis and post testing study of the netlist shows that:

1. *Damages* over VHDL can disappear due to register optimizing of the netlist, not due to the damage, due to the natural functional optimisation. The Leon2 that is being tested does not comprise all the functional features that are present in the Leon2. Due to this, in the final netlist the logic is reduced to those peripherals present.
2. *Damages* may not be observed due to "bad quality" of the test vectors to expose SEU faults, which would have had noticeable bad consequences with a different collection of test vectors.

Both cases have been observed in this test after a detailed inspection of the register database. The second case is particularly interesting because it offers a **new point of view** for using of the FT-UNSHADES

system. In the current case tests vectors coming from GR device production test, that assure a complete internal activation of all the internal device modules (as seen in the tests described in this report).

The tests have been re-launched concentrating the faults over the damaged modules, and even restricting the test to the damaged registers. In all of them the faults weren't seen.

2.5.2 Second Experience

In this test, the damages have been selected after a selection from the database obtained from the results of the 5 SEU experience. The corrupted voters have been selected from the database where two domains of the same registers were attacked and the fault detected at the outputs. In this case the selected fault is surely visible. A new VHDL library has been generated from the fault tolerant register library (called ftreg.vhd), called damftreg.vhd. Just substituting a particular fault tolerant register by its corresponding of the damaged library we easily can generate a new damaged Leon2-FT model.

Test 1. Fault in *icache* module.

The modified line is in the instruction cache module (ic module). In this case the value c.istate'length is 2.

```
rft05 : damftregv generic map (c.istate'length) -- '
        port map (clk, c.istate, r.istate);
```

The corresponding registers, followed by the actual number of SEU hits to them are represented as:

SEU_MUT/leon0/mcore0/proc0/c0/icache0/ftregs.rft05/rl.0.rft0/r(2) (13 times)
SEU_MUT/leon0/mcore0/proc0/c0/icache0/ftregs.rft05/rl.1.rft0/r(2) (30 times)

After a test of 200000 runs over the **entire netlist**, the fault has been detected, as expected, 43 times with the following numbers:

Test conditions: Leon2-FT-Dam1, 1 SEU per run, 200000 runs per test

Test#	T04-0-3
module	top
Registers	6463
Time (secs)	48625.1
Faults Det	43
Min (cycles)	2
Mean (cycles)	5.1
Max (cycles)	12
errorn	0

address	32
datao	11
dataen	0
ramsn	25
ramoen	22
rwen	0
romsn	0
iosn	0
oen	22
read	0
writen	0
pioo	0
pioen	0
Reset Mech	OK

Test 2. *ioport* module.

The modified lines are the input and output port (port module).

```
 rft02d : damftregv generic map (rin.pin1'length)  --'
              port map (clk, rin.pin1, r.pin1);
.......
 rft05d : damftregv generic map (rin.pout'length)  --'
              port map (clk, rin.pout, r.pout);
```

The corresponding faulty registers are represented in the internal database as explained in the following list, followed by the actual number of SEU hits to them.

SEU_MUT/leon0/mcore0/ioport0/regs1.rft05d/rl.0.rft0/r(2) (18 times)
SEU_MUT/leon0/mcore0/ioport0/regs1.rft05d/rl.1.rft0/r(2) (17 times)
SEU_MUT/leon0/mcore0/ioport0/regs1.rft05d/rl.2.rft0/r(2) (14 times)
SEU_MUT/leon0/mcore0/ioport0/regs1.rft05d/rl.4.rft0/r(2) (18 times)
SEU_MUT/leon0/mcore0/ioport0/regs1.rft05d/rl.6.rft0/r(2) (15 times)
SEU_MUT/leon0/mcore0/ioport0/regs1.rft05d/rl.7.rft0/r(2) (21 times)
SEU_MUT/leon0/mcore0/ioport0/regs1.rft05d/rl.9.rft0/r(2) (0 times)
SEU_MUT/leon0/mcore0/ioport0/regs1.rft05d/rl.10.rft0/r(2) (12 times)
SEU_MUT/leon0/mcore0/ioport0/regs1.rft05d/rl.11.rft0/r(2) (0 times)
SEU_MUT/leon0/mcore0/ioport0/regs1.rft05d/rl.12.rft0/r(2) (11 times)
SEU_MUT/leon0/mcore0/ioport0/regs1.rft05d/rl.13.rft0/r(2) (0 times)
SEU_MUT/leon0/mcore0/ioport0/regs1.rft05d/rl.15.rft0/r(2) (0 times)
SEU_MUT/leon0/mcore0/ioport0/r_3(2) (13 times)
SEU_MUT/leon0/mcore0/ioport0/r_4(2) (10 times)

SEU_MUT/leon0/mcore0/ioport0/r_5(2) (12 times)
SEU_MUT/leon0/mcore0/ioport0/r_6(2) (16 times)

In all cases the bit-flip produced “external action”, and the fault was detected.

After a test of 100000 runs over the **entire netlist**, the fault has been detected, as expected, 177 times with the following numbers:

Test conditions: Leon2-FT-Dam2, 1 SEU per run, 100000 runs per test

Test#	T05-0-3
module	Top
Registers	6463
time (secs)	27444,15
Faults Det	177
Min (cycles)	1
Mean (cycles)	1
Max (cycles)	1
errorn	0
address	0
datao	0
dataen	0
ramsn	0
ramoen	0
rwen	0
romsn	0
iosn	0
oen	0
read	0
writen	0
pioo	177
pioen	0
Reset Mech	OK

Test 3. *mctrl* module.

The modified lines are the mctrl module (possibly memory controller).

```
rft02d : damftregv generic map (rin.pin1'length)  --'
            port map (clk, rin.pin1, r.pin1);
```

The corresponding faulty registers are represented in the internal database as explained in the following list, joined with the actual number of hits to them.

SEU_MUT/leon0/mcore0/mctrl0/regs1.rft60d/rl.0.rft0/r(2) (6 times)

SEU_MUT/leon0/mcore0/mctrl0/regs1.rft60d/rl.1.rft0/r(2) (7 times)

In all cases the bit-flip produced external action, and the fault was detected.

After a test of 50000 runs over the **entire netlist**, the fault has been detected, as expected , 13 times with the following numbers:

Test conditions: Leon2-FT-Dam3, 1 SEU per run, 50000 runs per test

Test#	T06-0-3
module	Top
Registers	6463
time	14976.29
Faults Det	13
Min (cycles)	39
Mean (cycles)	81027
Max (cycles)	194382
errorn	0
address	0
datao	0
dataen	13
ramsn	13
ramoen	0
rwen	0
romsn	0
iosn	0
oen	0
read	0
writen	13
pioo	0
pioen	0
Reset Mech	OK

Test 4. Systematic attack for an internal submodule

Next test has been performed over the reset structure, because it seems to be critical, from the data obtained in test T00-8-0, T02-8-1 and T02-8-2, where few attacks over the subset of registers that compound the reset submoule have been attacked in a systematic way with 50000 attacks. Results are the following:

Test#	T07-8-3
module	Reset
Registers	15
time	11872
Faults Det	0

2.5.3 Conclusions

Corrupted netlist technique is a way to report that tests are made properly. Obviously to produce one netlist per FF is not possible but, to do it over several netlists provides confidence on the performed tests.

3. Conclusions

FT-UNSHADES system runs as expected, it emulates a circuit inside a radiation environment, and can find protection errors, if the test vector database is well prepared. The most important issue is that it's completely independent of who has designed the system and few requirements for test are needed. It provides deep analysis information about how the fault acts over the system. Fault testing can be addresses as desired, and can be directed to different parts of the circuit.

Reset strategy is not necessary to be done during the first clock period as required in the deliverables. Using the multi-SEU feature the reset strategy has been tested certifying the initialisation process. MasktoSEU command has been inserted to control the insertion process.

Test vector database quality has been tested through the voting-damage insertion technique, becoming FT-UNSHADES as a complement to radiation testing.

Design was intentionally corrupted in several points. In all cases the faults injected were detected (when the voting-damaged was chosen selectively!), but it's expected to find cases where the fault insertion does not manifest any activity at the output.

New tests over large IPs should be performed in order to qualify tests databases using FT-UNSHADES. Tests over protected IPs need to be launched over the non protected model, because it should give the idea of the percentage of faults potentially detected. In this case seems to be a 17% of the faults (1704 of 10000, see simple Leon2, single SEU, T00-0-0), this number is that *Quality of Test index* for the present design, being the reset module the most critical part.

4. Actions over TNT.

Large testing campaigns can be done if some commands are added for results management. The consequent action is to introduce command shell access to allow typical commands like: create directory, copy or move files, remove files,… etc.

The standard output (session.log and console) produces lots of unuseful data that can be reduced. Next generation of TNT tools will produce ‘dots’ in the console to certify its activity and session.log and will only show information of the faults.

5. Actions for going ahead with the FT-UNSHADES system

Next generation of FT-UNSHADES will concentrate the work on speed-up the test campaign. Current fault rate is approximately 4 faults per second. It's been proven that bottlenecks are basically due to the Windows structure. Windows takes about 10ms in context switching between USB communications and program activity. For this reason, the number of USB accesses should be reduced. The action should be to create in the CFPGA a hardware processor for fault injection, where faults locations are sent in packets of several faults and the processor produces the necessary actions automatically.

To do this some actions should be taken:

-Change the current model for FPGA (XS2C50) by a larger one (X2S150) what is immediate using a model pin to pin compatible.

-Design a low level bitstream processor in hardware

The design preparation flow should be improved to cover more VHDL coding structures. At the same time, the software toolbox should be integrated the tools in a graphic console that should control the preparation flow and display the commands to TNT tools.

Increase FT-UNSHADES design hosting capacity. This idea consists in the generation of a two boards structure, where the GOLD design runs in one board, MUT design in another and a synchronising mechanism for the two boards, performing the comparison between them.

Another important action is to guarantee that MUT is a RTL equivalent to the original ASIC netlist. The library matching techniques should be explored in detail.

SETs and Multi Bit Upsets (MBUs) are elements that should be analysed in order to produce models for their implementation in FT-UNSHADES.

It's been demonstrated that the study over the non-protected version of the circuit provides useful information about the quality of test vectors. We think that it's not strictly necessary to have both netlists, protected and non-protected. In the case of TMR protection it's possible to define a new testing mode, called "paired" where multi-bit faults are injected over two registers of the same TMR group. This is not immediate, because naming policy depends on many factors. For example, for Leon2-FT two registers of the same TMR are:

SEU_MUT/leon0/mcore0/reset0/regs1.rft00/r_3(0) SEU_MUT/leon0/mcore0/reset0/regs1.rft00/r_3(2)

While the method for TMR insertion of the XTMRtool is

SEU_MUT_uut/leon0_mcore0_reset0_rstout_4_TR0 SEU_MUT_uut/leon0_mcore0_reset0_rstout_4_TR1

Both naming conventions are completely different, so the new technique needs some effort.

Documents

Latest versions of TNT tools and Software Manual are 1.4.0 (Build date: Nov 2005) and 1.1 respectively.