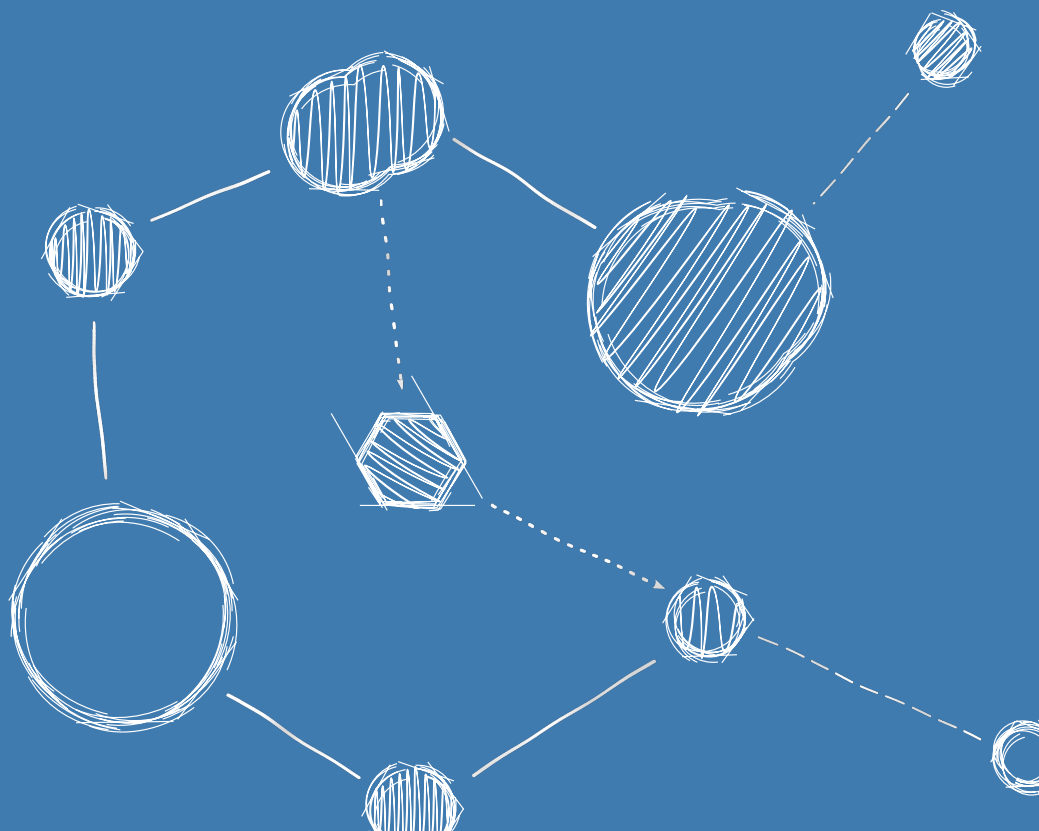


LEAN OS – AN OPERATING SYSTEM FOR THE SSDP

Test Specification



Test Specification

Reference: LEANOS-UVIE-TS-001

Version: Issue 1.0, May 1, 2016

Prepared by: Armin Luntzer¹

Checked by: Roland Ottensamer¹, Christian Reimers¹

Approved by: Franz Kerschbaum¹

¹ Department of Astrophysics, University of Vienna

Copyright ©2017

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Front-Cover, no Logos of the University of Vienna.

Contents

1	Introduction	5
1.1	Purpose of the Document	5
1.2	Test Overview	5
2	Applicable and Reference Documents	6
3	Terms, Definitions and Abbreviated Items	7
3.1	Acronyms	7
3.2	Glossary	7
4	Test Specification	10
4.1	Unit and Integration Tests	10
4.2	Hardware-Software Tests	11
4.3	Acceptance Tests	12
5	Detailed Test Definition	14
5.1	Hardware-Software Test Procedures	14
5.2	Hardware-Software Test Procedures	15
5.2.1	Xentium Processing Network	15
6	Test Plan Requirements to Specification Traceability	17

List of Specified Tests

D-UIT-0001	10
D-UIT-0002	10
D-UIT-0003	10
D-UIT-0004	10
D-UIT-0005	11
D-UIT-0006	11
D-HST-0001	11
D-HST-0002	11
D-HST-0003	12
D-HST-0004	12
D-SCC-0001	12
D-ACT-0001	12
D-ACT-0002	13

Revision History

Revision	Date	Author(s)	Description
0.1	01.05.2017	AL	Initial version
1.0	01.06.2017	FK	Document approved

1. Introduction

1.1 Purpose of the Document

The test plan for the the LeanOS operating system is defined in [\[1\]](#).

This test plan includes tests where the integrated operating system is subjected to tests in its operational environment.

This document specifies the acceptance tests for LeanOS.

A test is specified by defining the:

- pre-conditions for the test
- hardware baseline
- system configuration
- checks to be performed
- pass/fail criteria

1.2 Test Overview

A list of tests specified in this document is available in chapter: [List of Specified Tests](#)

2. Applicable and Reference Documents

- [1] *LeanOS Test Plan*. 2017.
- [2] *LeanOS User Manual*. 2017.

3. Terms, Definitions and Abbreviated Items

3.1 Acronyms

CPU	Central Processing Unit
DSP	Digital Signal Processor
ILP	Instruction Level Parallelism
MPPB	Massively Parallel Processor Breadboarding system
NoC	Network On Chip
SMP	Symmetric Multiprocessing
SoC	System On Chip
SSDP	Scalable Sensor Data Processor
VLIW	Very Long Instruction Word

3.2 Glossary

Central Processing Unit (**CPU**)

The Central Processing Unit is the electronic circuitry that interprets instructions of a computer program and performs control logic, arithmetic, and input/output operations specified by the instructions. It maintains high-level control of peripheral components, such as memory and other devices.

Digital Signal Processor (**DSP**)

A Digital Signal Processor is a specialised processor with its architecture targeting the operational needs of digital signal processing.

Instruction Level Parallelism (**ILP**)

Instruction-level parallelism (ILP) is a measure of how many instructions in a computer program can be executed simultaneously by the [CPU](#).

LEON2

The LEON2 is a synthesisable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture. It is highly configurable and particularly suitable for [System On Chip \(SoC\)](#) designs. Its source code is available under the GNU LGPL license

LEON3

The LEON3 is an updated version of the [LEON2](#), changes include [Symmetric Multiprocessing \(SMP\)](#) support and a deeper instruction pipeline

LEON3-FT

The LEON3-FT is a fault-tolerant version of the [LEON3](#). Changes to the base version include autonomous error handling, cache locking and different cache replacement strategies.

Massively Parallel Processor Breadboarding system (MPPB)

The Massively Parallel Processor Breadboarding system is a proof-of-concept design for a space-hardened, fault-tolerant multi-DSP system with various subsystems to build a powerful digital signal processing system with a high data throughput. Its distinguishing features are the [Network On Chip \(Network On Chip \(NoC\)\)](#) and the [Xentium DSPs](#) controlled by a [LEON2](#) processor. It was developed under ESA contract 21986 by Recore Systems B.V.

Network On Chip (NoC)

A Network On Chip is a communication system on an integrated circuit that applies (packet based) networking to on-chip communication. It offers improvements over more conventional bus interconnects and is more scalable and power efficient in complex [System On Chip \(SoC\)](#) designs.

Scalable Sensor Data Processor (SSDP)

The Scalable Sensor Data Processor (SSDP) is a next generation on-board data processing mixed-signal ASIC, envisaged to be used in future scientific payloads requiring high-performance on-board processing capabilities. It is built upon a heterogeneous multicore architecture, combining two [Xentium DSP](#) cores with a general-purpose [LEON3-FT](#) control processor in a [Network On Chip \(NoC\)](#).

SpaceWire

SpaceWire is a spacecraft communication network based in part on the IEEE 1355 standard of communications.

Symmetric Multiprocessing (SMP)

Symmetric Multiprocessing denotes computer architectures, where two or more identical processors are connected to the same periphery and are controlled by the same operating system instance.

System On Chip (SoC)

A System On Chip is an integrated circuit that combines all components of a computer or other electronic system into a single chip.

Very Long Instruction Word (VLIW)

Very Long Instruction Word is a processor architecture design concept that exploits [Instruction Level Parallelism \(ILP\)](#). This approach allows higher performance at a smaller silicon footprint compared to serialised instruction processors, as no instruction re-ordering logic to exploit superscalar capabilities of the processor must be integrated on the chip, but requires either code to be tuned manually or a very sophisticated compiler to exploit the full potential of the processor.

Xentium

The Xentium is a high performance [Very Long Instruction Word \(VLIW\) DSP](#) core. It operates 10 parallel execution slots supporting 32/40 bit scalar and two 16-bit element vector operations.

4. Test Specification

4.1 Unit and Integration Tests

D-UIT-0001	Identifier	Function
	Test Implementation	For each software component or module, a corresponding unit or integration test program is implemented by means of an adapted version of the <i>kselftest</i> framework. A shared collection of mockup and stub functions is maintained in order to facilitate testing.

Purpose | [R-UIT-0001](#), [R-UIE-0001](#)

D-UIT-0002	Identifier	Function
	Test Coverage	All implemented unit tests deliver 100% statement, decision and condition coverage. Any deviations are justified as part of the unit test implementation.

Purpose | [R-SCC-0004](#)

D-UIT-0003	Identifier	Function
	Test Execution	The collection of unit and integration tests is executable from the specified source directory via the <i>make</i> command.

Purpose | [R-UIT-0003](#)

D-UIT-0004	Identifier	Function
	Test Results	The test executables generate human readable text output summarising the number, types and outcome of each test performed.

Purpose | [R-UIT-0004](#)

D-UIT-0005	Identifier	Function
	Auto-mated Testing	The test executables report failed tests as shell exit codes, so the outcome can be detected by an external test program such as <i>git bisect</i> .

Purpose | [R-UIT-0005](#)

D-UIT-0006	Identifier	Function
	Test Suite	All unit and integration tests, along with stub and mockup functions are stored in the LeanOS source tree directory <i>tools/testing/unittest/</i> .

Purpose | [R-UIT-0002](#), [R-UIE-0001](#)

4.2 Hardware-Software Tests

D-HST-0001	Identifier	Function
	Test Implementation	For each software component or module that needs verification in a hardware environment, a test program is implemented.

Purpose | [R-HST-0001](#)
Comment | Since these tests typically involve more complex control via other tools (such as *grmon* or [SpaceWire](#) interfaces), they are implemented as shell scripts that set up the required configuration and executes the test on the target hardware.

D-HST-0002	Identifier	Function
	Test Suite	All hardware-software tests are stored in the LeanOS source tree directory <i>tools/testing/hwtests/</i> .

Purpose | [R-HST-0002](#)

D-HST-0003	Identifier	Function
	Test Results	The test executables generate human readable text output summarising the number, types and outcome of each test performed.

Purpose | [R-HST-0003](#)

D-HST-0004	Identifier	Function
	Automated Testing	The test executables report failed tests as shell exit codes, so the outcome can be detected by an external test program such as <i>git bisect</i> .

Purpose | [R-HST-0004](#)

D-SCC-0001	Identifier	Function
	Code Coverage	Statement, decision and condition coverage is determined via <i>gcov</i> . A <i>make</i> target <i>coverage_test</i> is available in the <i>Makefile</i> of the unit test directory of the LeanOS source tree. Human-readable reports are placed in the respective subdirectories for each individual unit test.

Purpose | [R-SCC-0001](#), [R-SCC-0002](#), [R-SCC-0003](#)

4.3 Acceptance Tests

D-ACT-0001	Identifier	Function
	Acceptance Test Definition	Any acceptance test procedures are defined in chapter Detailed Test Definition of this document.

Purpose | [R-ACT-0001](#)

D-ACT-0002	Identifier	Function
	Accep- tance Test Use Cases	The configuration and operational environment of LeanOS is defined for each acceptance test of an identified use case.
Purpose	R-ACT-0002	

5. Detailed Test Definition

5.1 Hardware-Software Test Procedures

The following tables list the tests specified in this document. Note that in the present version of the document, this list is *incomplete*, as the specification of the [SSDP](#) has not yet been released at this time.

Procedural steps are presented in tabular form and numerical sequence where applicable. Column headings are *TYPE*, *SEQ*, *Description* and *VERIFY*.

Procedural step *types* are:

CMT	Comment
PRE	Pre-condition
PST	Post-condition
STP	Step

SEQ indicates the sequence number of a step.

Description details the purpose and procedure of a step.

VERIFY lists any requirements or specifications verified in a step (if applicable).

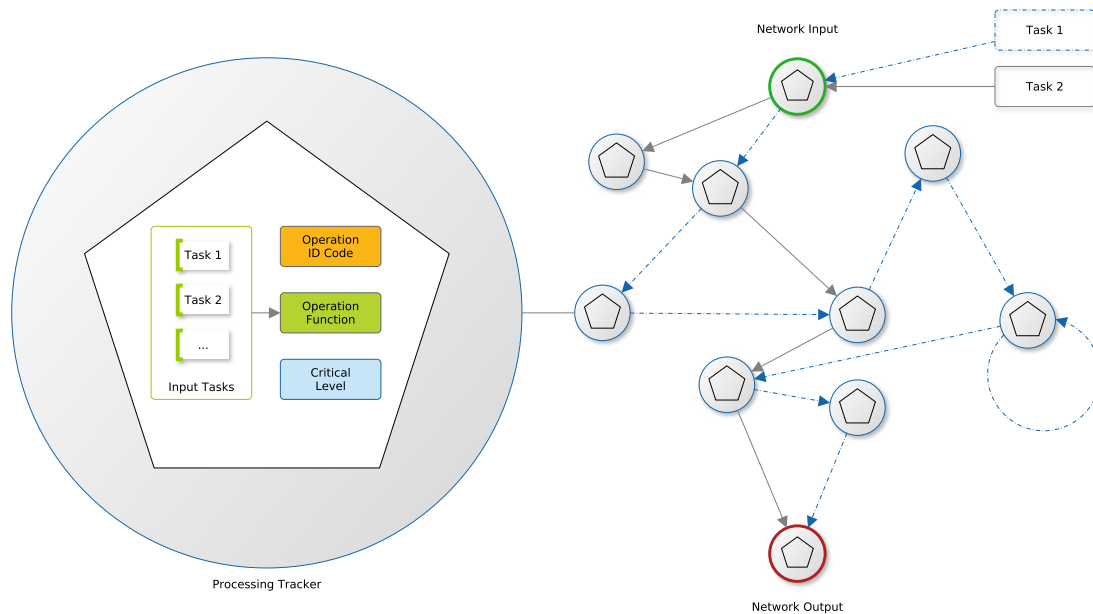


Figure 5.1: Tasks in the processing network propagate according to the operational code identifiers in their processing task list. For details, see [2].

5.2 Hardware-Software Test Procedures

5.2.1 Xentium Processing Network

This is an integrated test of the **Xentium** Processing Network (see Figure 5.1). It loads a set of **Xentium** program kernels in the Xentium driver of LeanOS, which are then used to apply processing operations on a series of input tasks to simulate a processing pipeline.

The Xentium kernels used in this test are:

- deglitch** applies a deglitching operation
- dummy** a simulation dummy that just passes on a task
- stack** collects multiple tasks and stacks their data into a single task
- rampfit** fits a linear function to a given number of samples

The test generates a total of 12 tasks (data sets) as input in groups of 4 tasks of 32, 64 and 64 samples of incremental value, containing a single glitch per data set. The stacking parameter is set to 4 and the ramp length is set to 32 samples.

The network then processes the tasks. Given 12 inputs, a stacking of 4 and a ramp length of 32, the network outputs 1 data point output for the first 4 task inputs of 32 samples and 2 data

points for inputs of 4x64 samples each, i.e. 5 data points in total. If this result is obtained, the test is considered a success.

ID	HWT-0001		
Purpose	Integrated System Test of Xentium Processing Network		
TYPE	SEQ	Description	VERIFY
CMT		This is a test to verify the function of the Xentium Processing Network implementation by generating a series of processing tasks and running them through the processing network.	
PRE		a MPPB ver. 2.x hardware model	
PRE		<i>grmon</i> ver. 1.x	
PRE		Ensure connection of MPPB to test control host via serial cable on DEBUG UART to virtual device <code>/dev/ttyS0</code> on the host platform.	
PRE		A checkout of the LeanOS source directory.	
PRE		MPPB/Xentium toolchain installed and configured	
STP	1	Ensure that the MPPB is powered and reset.	
STP	2	Execute test script in the <code>tools/testing/hwtests/xen_proc_net</code> subdirectory of the LeanOS source tree.	
STP	3	Verify that the program image is built.	
STP	4	Verify that the program image is uploaded to the MPPB via <i>grmon</i>	
STP	5	Verify that the program is started on the MPPB .	
STP	6	Verify that the program reports SUCCESS on termination.	
CMT		Test completed.	

