VERIFICATION PLAN

32-bit μDLX Core Processor

Universidade Federal da Bahia

Version: 1.0

GNU LGPL License

Revision History

Date	Description	Author(s)
05/23/2014	First Verification Plan version.	Victor Valente
06/07/2014	Adding information to Overview section.	Lauê Rami and Igo Amauri

CONTENTS

1 Introduction **4**

1.1 Document Purpose 4

1.2 Stakeholders 4

1.3 Document Outline Description 4

1.4 Acronyms and Abbreviations 4

2 DUT Overview **5**

3 Verification Environment **9**

3.1 Design Under Test Interface (DUT_IF) 9

3.2 Monitor 9

3.2.1 Forwarding 10

3.2.2 Branch Taken 10

3.2.3 Load Hazard 10

3.3 Verification Environment Design Specification 10

4 Features List **11**

5 Test List **12**

6 Assertions **14**

7 Coverage and Completion Requirements **15**

8 Resource Requirements **16**

9 Schedule **17**

1. Introduction

1.1. Document Purpose

The purpose of this document is define the verification plan of the uDLX Implementation. This document includes the verification environment used to perform the verification of the processor, beside the main characteristics of the design, the list of tests, list of assertions, and others.

1.2. Stakeholders

Name	Roles/Responsibilities
Igo Amauri Luz	TBD
Lauê Rami Costa	TBD

1.3. Document Outline Description

1.4. Acronyms and Abbreviations

Acronym	Description
ASIC	Application Specific Integrated Circuit
DUT	Design Under Test

2. DUT Overview

The uDLX is a simplified architecture of the DLX processor. The uDLX correspond to a RISC general propose microprocessor, memory access, arithmetic and logical, and branch instructions. The processor has 5 stages in a pipeline architecture that are : instruction fetch, instruction decode, execute, memory access, and write back.

The uDLX architecture interface is composed by the following components:

- uDLX 32-bit Core: The core four-deep pipeline processor.
- Memory Interface: Provides a middle layer between the core processor and the external memories. This interface also controls the bootloader process.
- SDRAM Controller: Provides the interface for controlling the external SDRAM.
- SRAM Controller: Provides the interface for controlling the external SRAM.

The Figure 1 shows the architecture interface.

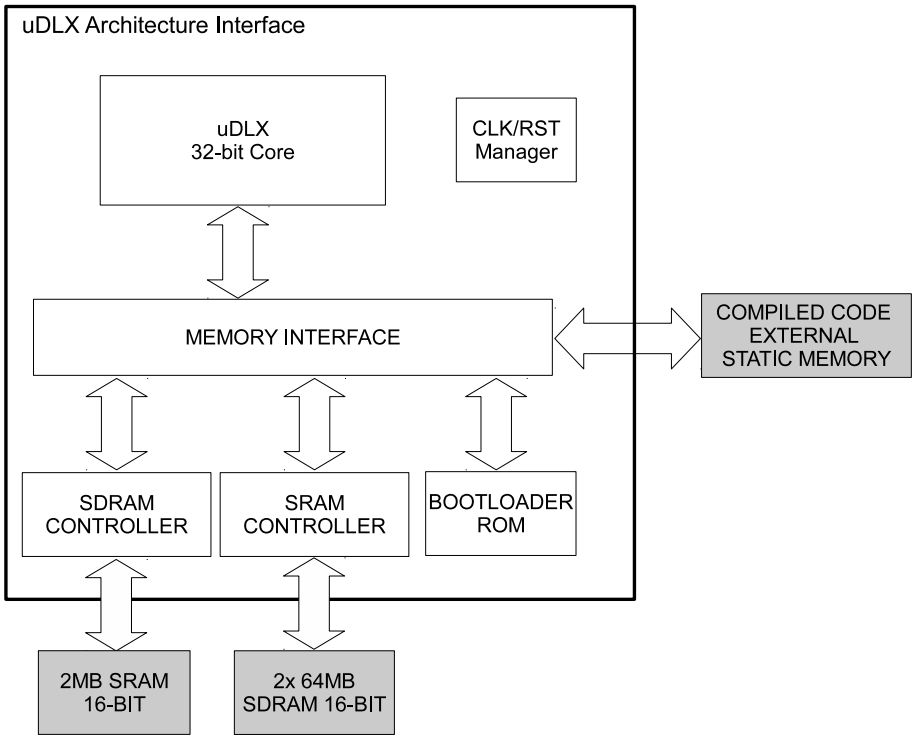

Figure 1: μDLX Architecture Interface.

A Harvard architecture is implemented using a SRAM with 16 bits of data width to access the instructions and a SDRAM to access data. In order to avoid stall due to time need to access the memories the processor and memories controllers work in different clock rate.

The main features of the uDLX are:

- uDLX with multiple stages
- Hazard handling

- SRAM memory access
- SDRAM memory access

The Figure 2 shows the Top Architecture of the μDLX Processor.

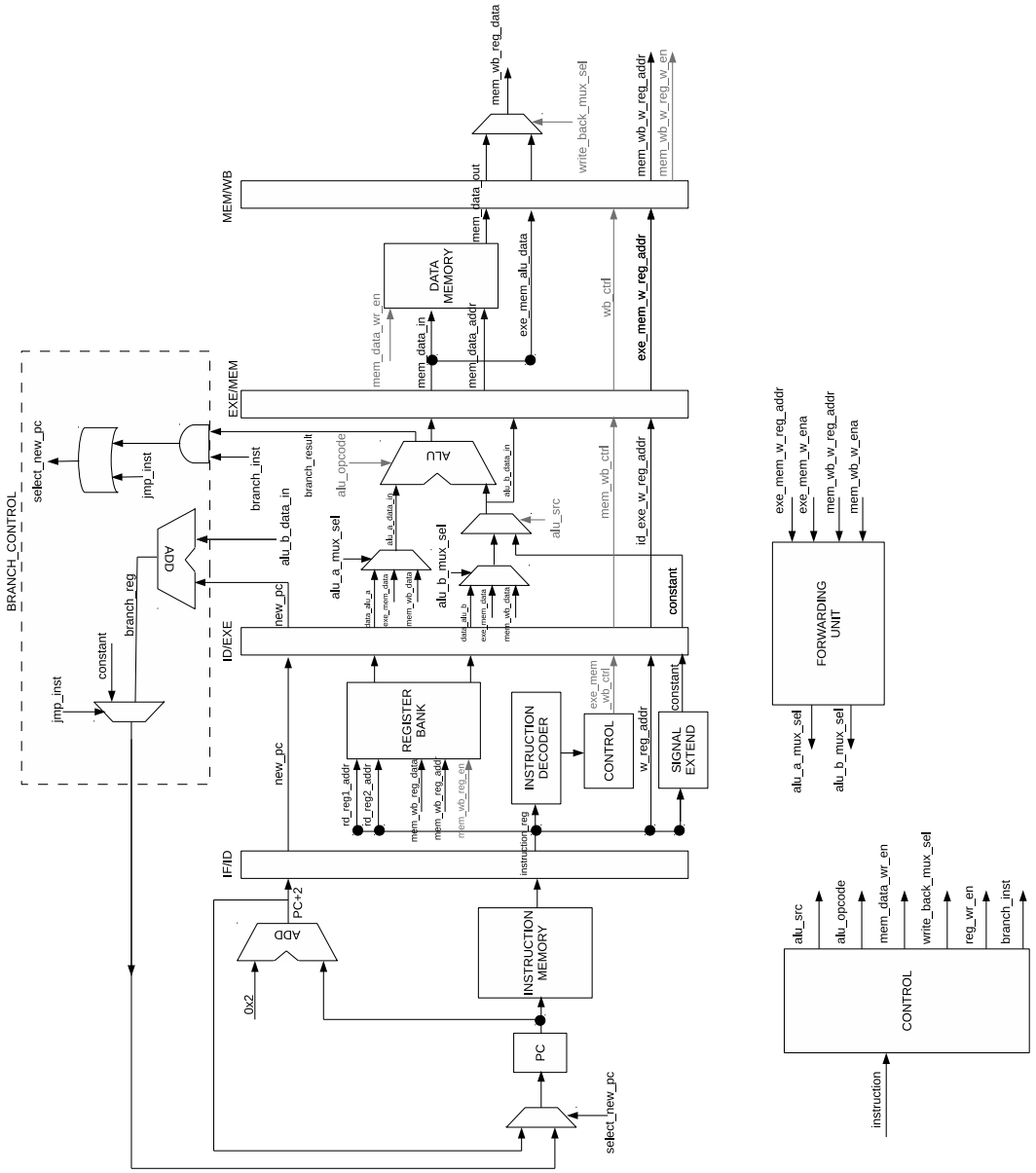

Figure 2: μDLX top level architecture overview.

In order to be the most close to the DLX and also be as compatible as possible to MIPS R2000 and R3000 architecture few instructions were added to the instruction set of this processor. The instruction set supported in the project are: I-type, R-type and J-type. The Figure 3 shows the types of instructions.

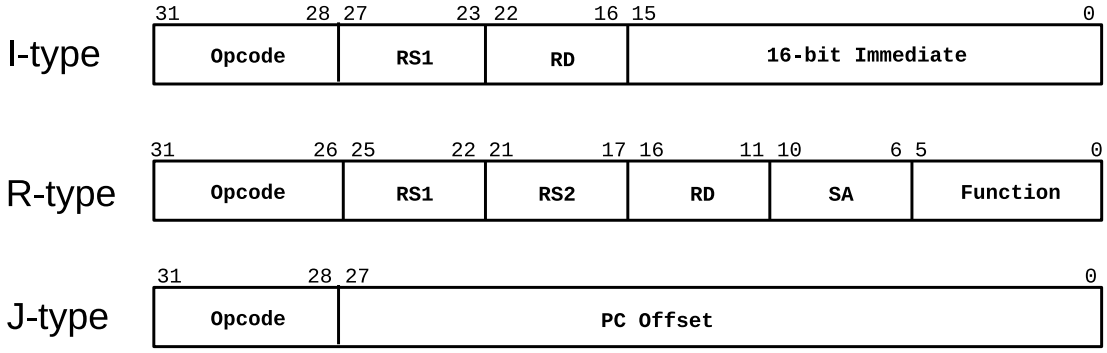

Figure 3: μDLX Instruction Set types definitions.

3. Verification Environment

The verification methodology is based in a simple testbench. Part of the verification will be done using waveform based verification. Some special situations will be verified using assertion based verification. The design under test (DUT) interface will be responsible to catch data from the uDLX and send to the monitor that will have all assertions. The following topics will describe the items that compose the Verification Plan. The Figure 4 shows the Verification Environment.

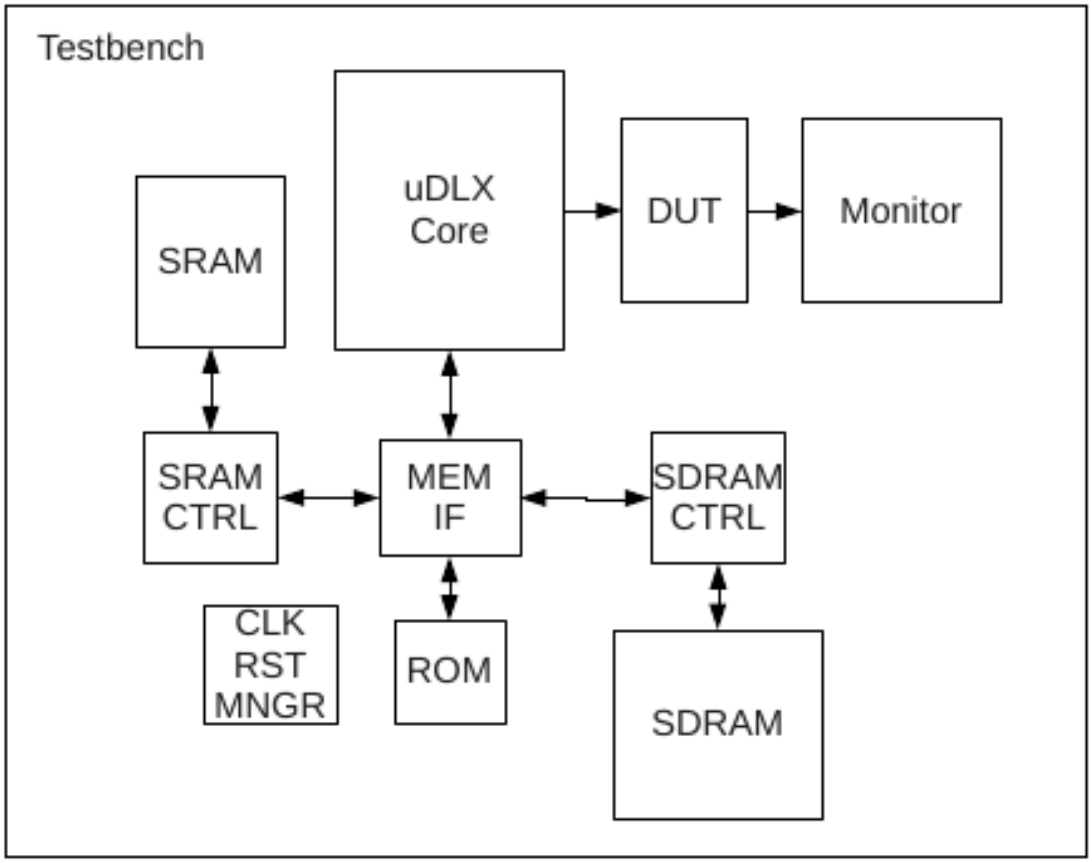

Figure 4: μDLX Verification Environment.

3.1. Design Under Test Interface (DUT_IF)

The Design Under Test Interface (DUT_IF) makes the interface between the monitor and the DUT. The DUT_IF is responsible to control the data that are transmitted between the Verification Environment and DUT.

3.2. Monitor

The monitor is responsible to observe the behavior of the DUT and, also, collect the output to verify if the instructions are working as expected. The main forwarding, branch taken, and load hazard. The monitor have all assertions that will ensure if the behavior of the DUT is correct.

3.2.1. *Forwarding*

The forwarding operation can occur between the execution and different blocks. This assertion will be used to ensure the forwarding unit is activating the correct signal and doing what is expected.

3.2.2. *Branch Taken*

As the uDLX developed work in the branch not taken method, the monitor needs to ensure that when a branch is taken all flush and new address will work correctly.

3.2.3. *Load Hazard*

During a load hazard the uDLX stall and flush specific stages. The assertions will monitor this condition and give warnings for any unpredicted behavior.

3.3. Verification Environment Design Specification

Component	Description
Document Name and Version	μDLX Verification Plan
Document Version and Date	Version 1.0, June 23, 2014
Author(s) / Owner(s)	Lauê Rami / Igo Amauri
Verification Methodology	Top-Down
Verification Methods	Simulation and Formal Verification
Simulation Components	
Application	Cadence Incisive
Languages	System Verilog
Verification Environmentv	Custom testbench
Testbench files	At directory:
Formal Verification Components	Cadence Encounter Conformal
Application	
Technologies	FPGA DE2-115

4. Features List

The features list, describes the features that are planned to be implemented.

Feature Number	Feature Description	Priority
DLX_F1	Instruction must activate the signals corresponding to the functionality	10
DLX_F2	Forwarding Unit should identify and forward data correctly	8
DLX_F3	During a load hazard the stall should work correctly	8
DLX_F4	Jump hazard should flush before instructions and change PC	8
DLX_F5	Communication with SRAM must work properly	9
DLX_F6	Read and write operation to the SDRAM	9
DLX_F7	The registers of the Register Bank must be reading and writing	10
DLX_F8	All interfaces protocols must work properly	9

5. Test List

The test list define the tests that will be performed with the DUT to ensure that the behavior is correct. In addition, this tests list ensures achieving coverage levels.

Test Number	Description	Method	Level	Features Verified	Priority	Owner	Completion
DLX_T1	Execution of all instructions of the Arithmetic category.	Sim	Unit	DLX_F1	5	Lauê	0%
DLX_T2	Execution of all instructions of the Data Transfer category.	Sim	Unit	DLX_F1	5	Lauê	0%
DLX_T3	Execution of all instructions of the Logical category.	Sim	Unit	DLX_F1	5	Lauê	0%
DLX_T4	Execution of all instructions of the Conditional Branch category.	Sim	Unit	DLX_F1	5	Lauê	0%
DLX_T5	Execution of all instructions of the Unconditional Jump category.	Sim	Unit	DLX_F1	5	Lauê	0%
DLX_T6	Test forwarding unit in all possible situations	Sim	Unit	DLX_F1, DLX_F2	5	Igo	0%
DLX_T7	Test the load hazard	Sim	Unit	DLX_F1, DLX_F3	4	Igo	0%
DLX_T8	Test jump and branch instructions	Sim	Unit	DLX_F4	8	Igo	0%
DLX_T9	Test the SDRAM access	Sim	Unit	DLX_F6	7	Lauê	0%
DLX_T10	Test the SRAM access	Sim	Unit	DLX_F5	9	Lauê	0%

continued on next page

continued from previous page

Test Number	Description	Method	Level	Features Verified	Priority	Owner	Completion
DLX_T11	Run programs with the complete architecture	Sim	Unit	DLX_F5, DLX_F6	8	Igo	0%
DLX_T12	Test All interfaces protocols	Sim	Unit	DLX_F8	8	Igo	0%

6. Assertions

Assertion will be used to verify the instructions, specially the instructions that have stall, forwarding and flush.

Date	Criteria	Status
06/25/2014	Assertion to the correct fecth_top operation	Not started yet
06/25/2014	Assertion to verify the decode operation	Not started yet
07/02/2014	Assertion for the execution block operation	Not started yet
07/02/2014	Assertion for the memory read and write back	Not started yet
07/02/2014	Assertion to forwarding operation	Not started yet
07/02/2014	Assertion during branches and jumps	Not started yet
07/02/2014	Assertion during load hazard	Not started yet
07/02/2014	Assertion to verify interfaces protocols	Not started yet

7. Coverage and Completion Requirements

Date	Criteria	Required Level	Current Level	Status
06/25/2014	Block coverage	98%	0%	Not started yet
06/25/2014	Expression coverage	95%	0%	Not started yet
06/25/2014	Toggle coverage	90%	0%	Not started yet
06/25/2014	Code coverage	95%	0%	Not started yet
06/25/2014	State coverage on FSM	100%	0%	Not started yet
06/25/2014	Arc coverage on FSM	100%	0%	Not started yet
06/25/2014	Functional coverage	100%	0%	Not started yet
06/25/2014	Bug report rate for SPI	< 1 bug/day	-	Not started yet

8. Resource Requirements

Resource	Quantity	Description	Start	Duration
Engineering Resources				
Verification Engineer	2	1 year experience	06/25/14	10 days
Compute Resources				
Computer	1		06/25/14	10 days
Software Resources				
Candence Incisive	1		06/25/14	10 days
ALTERA Quartus	1	WEB Edition	06/25/14	10 days
ALTERA ModelSIM	1	WEB Edition	06/25/14	10 days

9. Schedule

Resource	Start	Duration	Action	Compute Resource
Lauê Rami	06/20/14	2 Days	Develop Verification Test Plan	N/A
Igo Amauri	06/22/14	1 Day	Review & Approve Test Plan	N/A
Lauê Rami	06/25/14	5 Days	Create Verification Environment	Server 1
Igo Amauri	06/28/14	3 Days	Features F1,F2,F3, and F4 Verification	Server 1
Igo Amauri	07/02/14	4 Days	Add assertion and code coverage	Server 1
Lauê Rami	07/10/14	2 Days	Complete System simulation and functional verification	Server 1