

# **CPU Architecture**

**LAB3 assignment**

**Digital System Design with VHDL  
(Simple RISC Multi-Cycle CPU design)**

**Hanan Ribo**

**23/04/2025**

## Table of contents

1.	Aim of the Task: .....	3
2.	Assignment definition: .....	3
3.	Controller based system: .....	3
4.	File-based simulation of DUT: .....	8
5.	Applications and Software Quality Assurance: .....	8
6.	SRMC Toolchain IDE: .....	8
7.	Requirements: .....	9
8.	Grading Policy .....	10

## 1. Aim of the Task:

- System design using concurrent and sequential logic principles using advanced Simulation methods (based on material given in LAB1 and LAB2 tasks).
- Controller design based on methodology of Control and Datapath separation.
- Preparation for LAB4 task – FPGA based design synthesis of a given design.
- Proper analysis and understanding of architecture design.
- Performing and understanding functional validation and verification of an architecture design.

## 2. Assignment definition:

In this task, you are required to design a controller-based processing machine as a Multi-cycle CPU to run a given program code. The preparation material for this lab has been learned in the prerequisites course “Central Processing Unit Architecture - theory” until lecture five and includes in addition to the self-learning material given on a subjects of FSM methodology design and advanced functional verification.

## 3. Controller-based system:

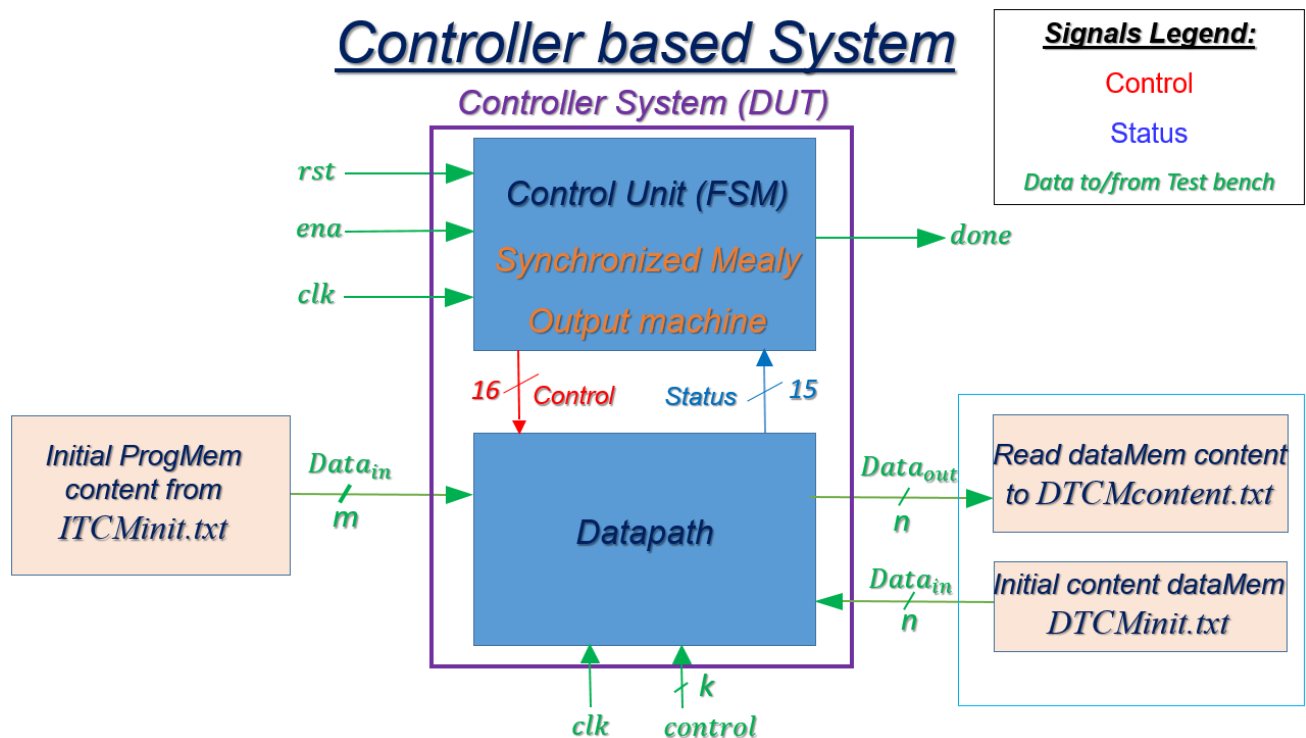


Figure 1: Overall DUT structure

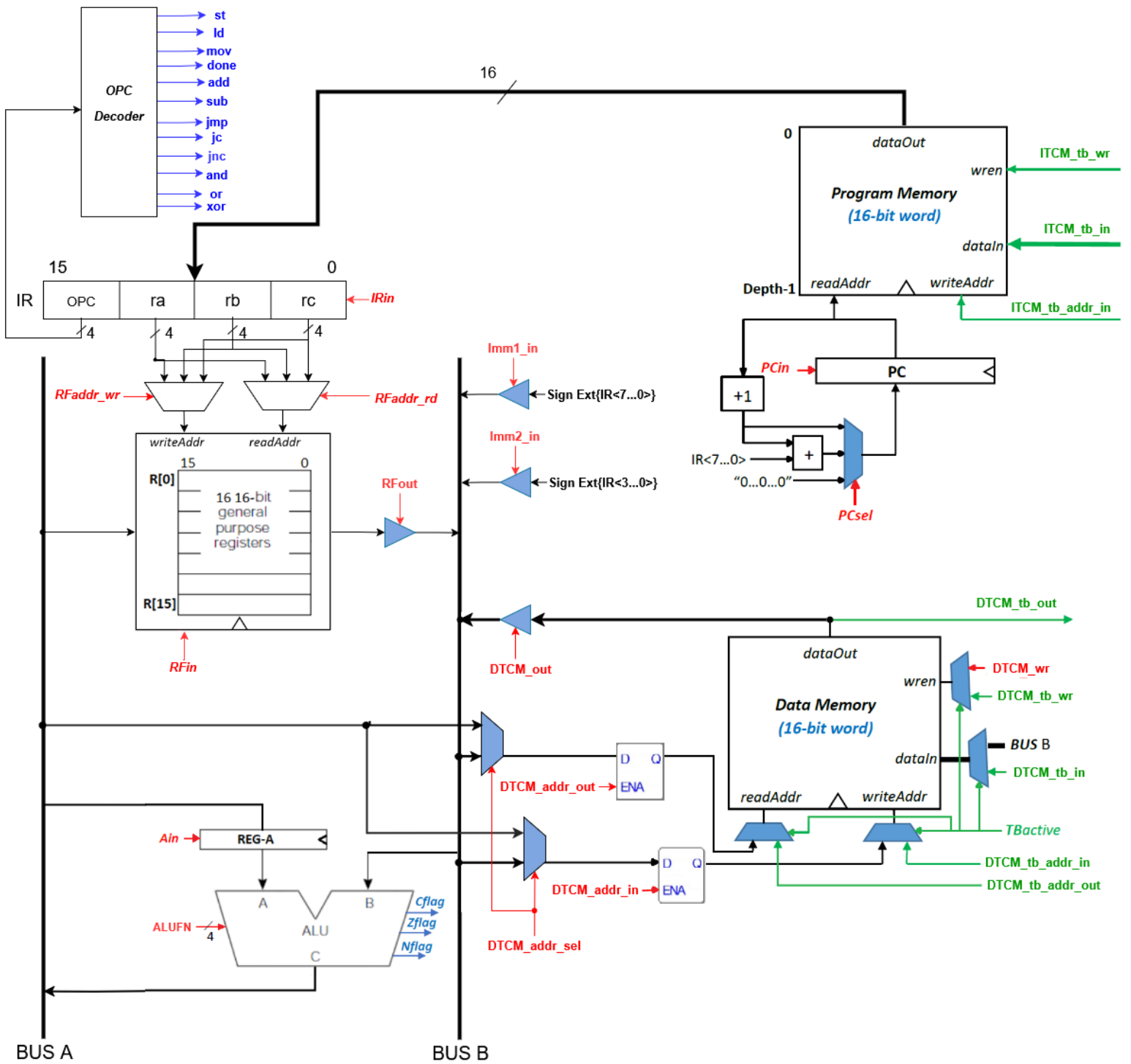


Figure 2: SRMC-I Datapath structure of a 2-BUS multicycle CPU

# Control Unit

**Signals Legend:** Control, Status, Data to/from Test bench

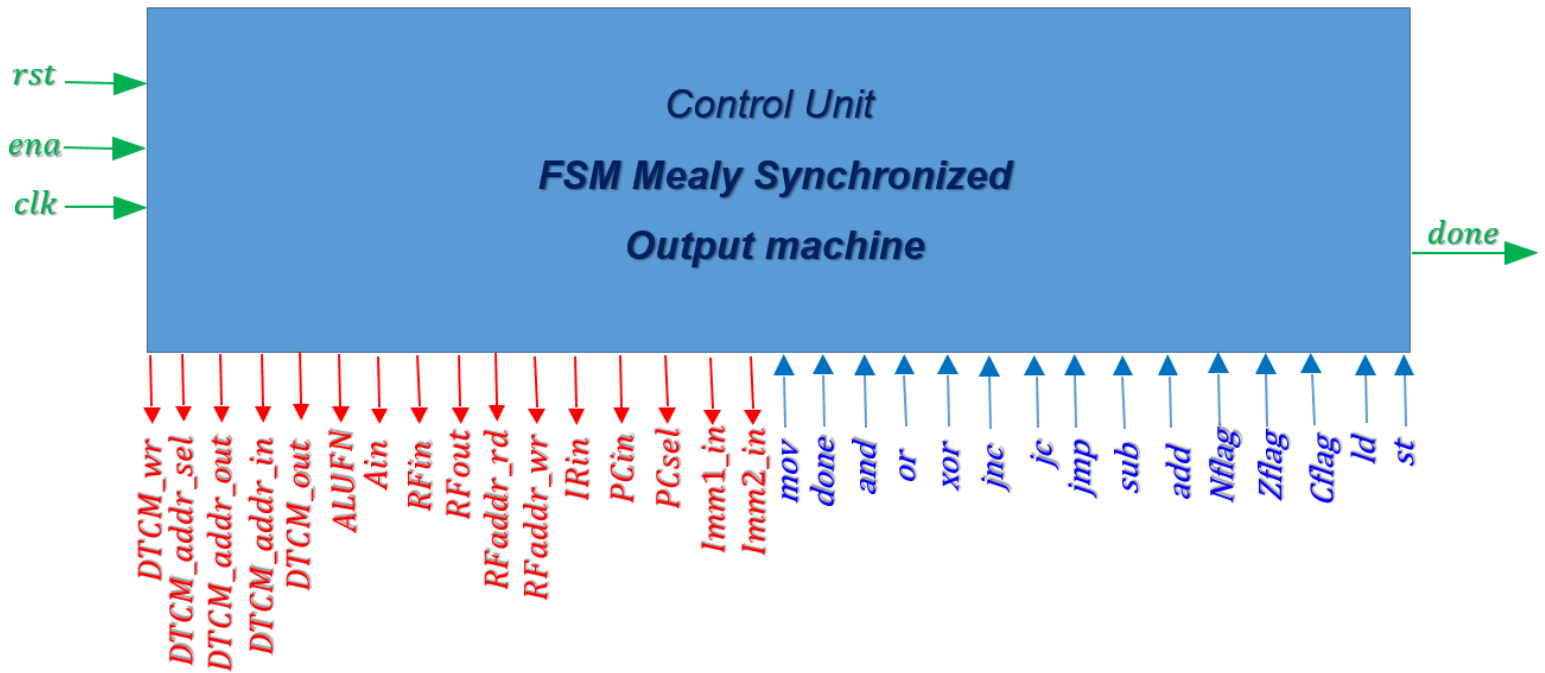


Figure 3: SRMC-I Control unit module

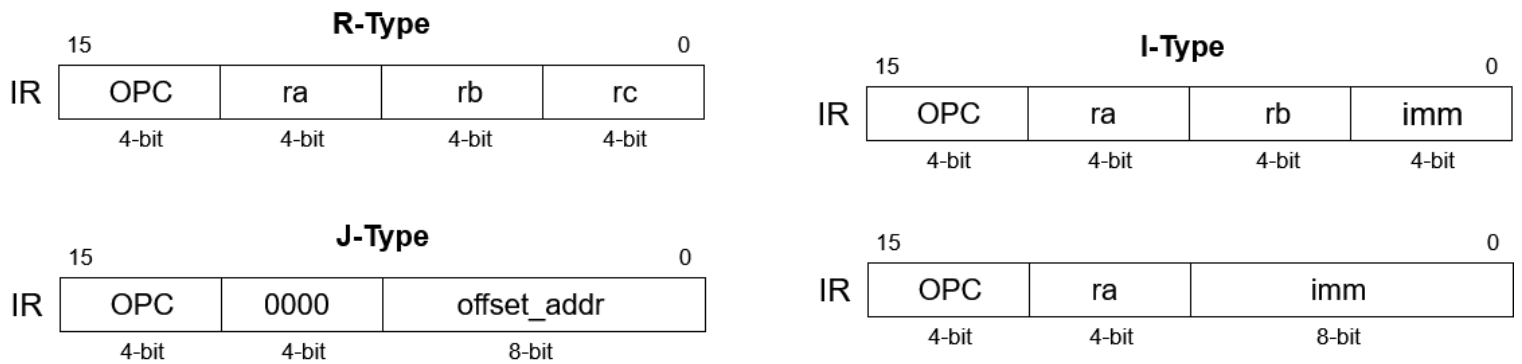


Figure 4: SRMC-I Instruction Formats subdivided into three types

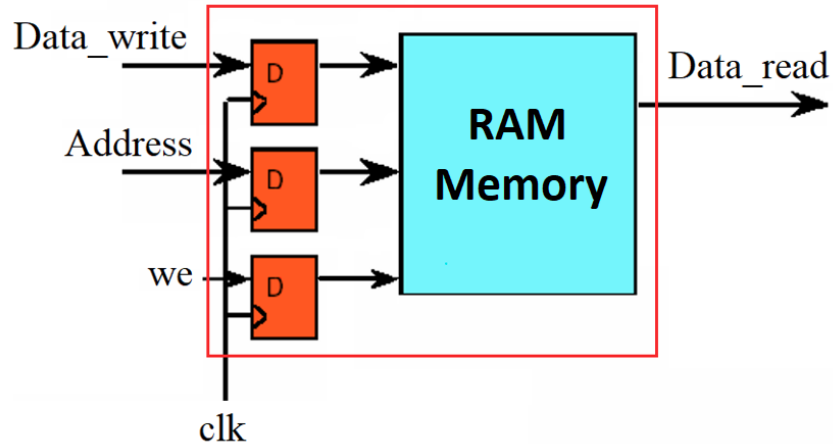
ADDRESS MODE	SYNTAX	INSTRUCTION FORMAT	EXAMPLE	OPERATION
Register	add ra,rb,rc	R-Type	add r4,r2,r1	$R[4] \leftarrow R[2] + R[1]$
Immediate	mov ra,imm	I-Type	mov r5,0x30 mov r5,Var	$R[5] \leftarrow 0x30$ $R[5] \leftarrow \text{Var}$
Direct	ld ra,imm(r0)		ld r4,0x20(r0) ld r4, Var(r0)	$R[4] \leftarrow M[0x20]$ $R[4] \leftarrow M[\text{Var}]$
Indirect	ld ra,0(rb)		ld r4,0(r3)	$R[4] \leftarrow M[R[3]]$
Indexed	ld ra,imm(rb)		ld r4,0x20(r3) ld r4, Var(r3)	$R[4] \leftarrow M[0x20 + R[3]]$ $R[4] \leftarrow M[\text{Var} + R[3]]$

**Table 1: SRMC-I Addressing Mode**

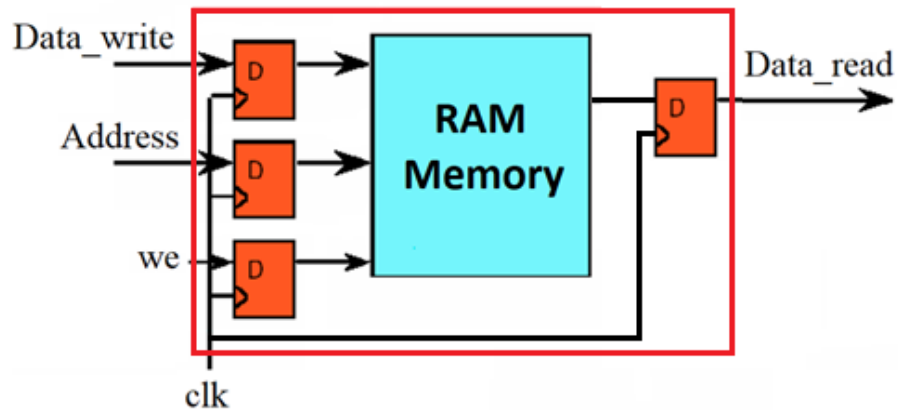
Instruction Format	Decimal value	OPC	Instruction	Explanation	N	Z	C
R-Type	0	0000	add ra,rb,rc nop	$R[\text{ra}] \leftarrow R[\text{rb}] + R[\text{rc}]$ $R[0] \leftarrow R[0] + R[0]$ ( <i>emulated instruction</i> )	*	*	*
	1	0001	sub ra,rb,rc	$R[\text{ra}] \leftarrow R[\text{rb}] - R[\text{rc}]$	*	*	*
	2	0010	and ra,rb,rc	$R[\text{ra}] \leftarrow R[\text{rb}] \text{ and } R[\text{rc}]$	*	*	-
	3	0011	or ra,rb,rc	$R[\text{ra}] \leftarrow R[\text{rb}] \text{ or } R[\text{rc}]$	*	*	-
	4	0100	xor ra,rb,rc	$R[\text{ra}] \leftarrow R[\text{rb}] \text{ xor } R[\text{rc}]$	*	*	-
	5	0101	<i>unused</i>				
	6	0110	<i>unused</i>				
J-Type	7	0111	jmp offset_addr	$\text{PC} \leftarrow \text{PC} + 1 + \text{offset\_addr}$	-	-	-
	8	1000	jc /jhs offset_addr	If(Cflag==1) $\text{PC} \leftarrow \text{PC} + 1 + \text{offset\_addr}$	-	-	-
	9	1001	jnc /jlo offset_addr	If(Cflag==0) $\text{PC} \leftarrow \text{PC} + 1 + \text{offset\_addr}$	-	-	-
	10	1010	<i>unused</i>				
	11	1011	<i>unused</i>				
I-Type	12	1100	mov ra,imm	$R[\text{ra}] \leftarrow \text{imm}$	-	-	-
	13	1101	ld ra,imm(rb)	$R[\text{ra}] \leftarrow M[\text{imm} + R[\text{rb}]]$	-	-	-
	14	1110	st ra,imm(rb)	$M[\text{imm} + R[\text{rb}]] \leftarrow R[\text{ra}]$	-	-	-
Special	15	1111	done	Signals the TB to read the DTCM content	-	-	-

Note: \* The status flag bit is affected , - The status flag bit is not affected

**Table 2: SRMC-I instruction set, recall that R[0] is a zero value (hard-wired).**

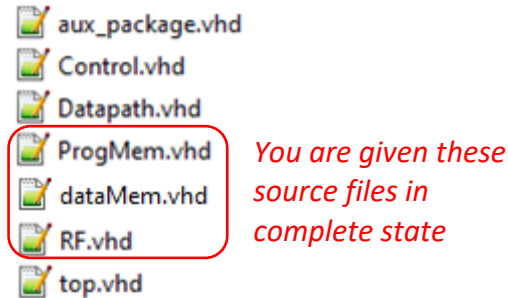


**Figure 5: Single Port RAM with an unregistered output. This structure is used for Register File and Program Memory (ITCM) with a given VHDL code.**



**Figure 6: Single Port RAM with a registered output. This structure is used for Data Memory (DTCM) with a given VHDL code.**

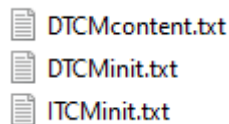
- **The next encircled files must be used in your project:**



- ✓ You must use the listed above source files, you are required to fill them up yourself.
- ✓ You can add additional VHDL source code files.
- ✓ Other entities can be designed and modeled behaviorally, structurally, etc.

#### 4. File-based simulation of DUT:

- As depicted in Figure 1, we use a file-based simulation to simulate the system DUT.
- Two Input and one Output files (according to the above given figure in clause 3):



#### 5. Applications and Software Quality Assurance:

Each application in the next list contains five files:

C file, ASM file, two binary files for ITCM and DTCM initialization, and a file of DTCM content after code running.

- ✓ [Code Example 1](#)
- ✓ [Code Example 2](#)
- ✓ [Code Example 3](#)
- ✓ [Code Example 4](#)

#### 6. SRMC Toolchain IDE:

The SRMC Toolchain contains, for now, only a compiler; the rest should be done manually.

[SRMC-I compiler v1\\_1.exe \(for Windows\)](#)

[SRMC-I compiler v1\\_1- python code](#)

[SRMC-I Compiler\\_Emulator\\_Simulator\\_v2.exe](#)

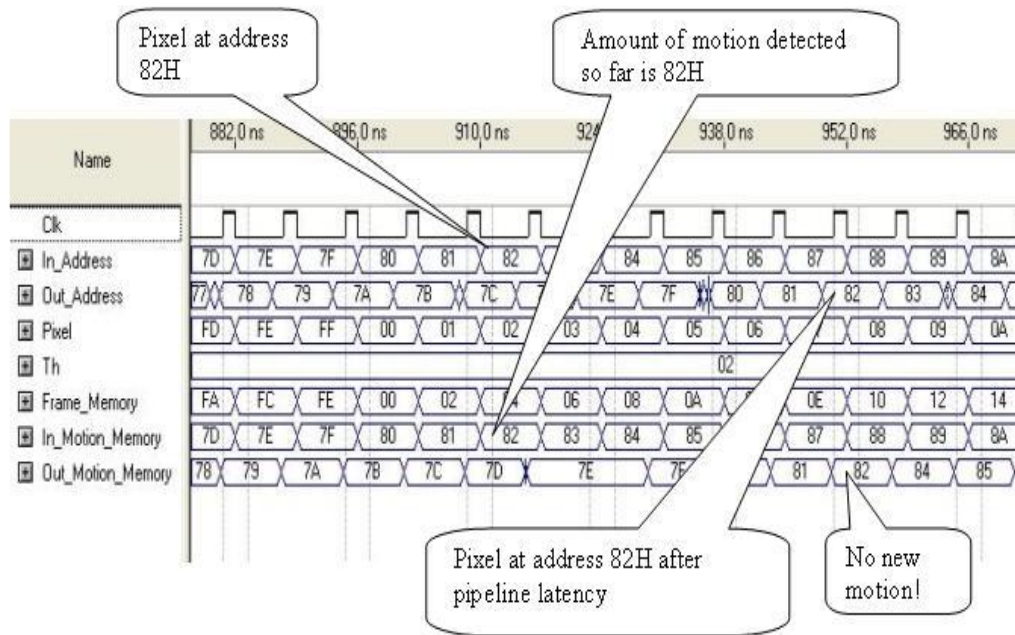


## 7. Requirements:

1. The design must be well commented on.
2. Elaborated analysis and waveforms:
  - Remove irrelevant signals.
  - Zoom on regions of interest.
  - Draw clouds on the waveform with explanations of what is happening (Figure 4).
  - Change the waveform colors in ModelSim for clear documentation  
(Tools->Edit Preferences->Wave Windows).
3. A ZIP file in the form of **id1\_id2.zip** (where id1 and id2 are the identification number of the submitters, and id1 < id2) *must be upload to Moodle only by student with id1* (any of these rule's violation disqualifies the task submission).
4. The **ZIP** file will contain (*only the exact next sub folders*):

Directory	Contains	Comments
DUT	Project VHDL files	<b>Only VHDL files of DUT</b> , excluding test bench <b>Note: your project files must be well compiled without errors as a basic condition before submission</b>
TB	VHDL files that are used for test bench	One <b>tb.vhd</b> for the Datapath One <b>tb.vhd</b> for the Control unit One <b>tb.vhd</b> for the overall DUT
SIM	ModelSim DO files (wave, list)	One <b>tb.vhd</b> for the Datapath One <b>tb.vhd</b> for the Control unit One <b>tb.vhd</b> for the overall DUT (with Datapath and Control unit signals included)
DOC	Project documentation	<ul style="list-style-type: none"> <li>• <b>readme.txt</b> (list of the DUT *.vhd files with their brief functional description)</li> <li>• <b>pre3.pdf</b> (report file that includes brief explanation of the top module with its wave diagrams as shown in figure 4)</li> <li>• <b>designGraph.pdf</b> (elaborated FSM graph of your DUT)</li> </ul>

Table 2: Directory Structure



**Figure 4: Clouds over the waveform example**

## 8. Grading Policy

Weight	Task	Description
10%	Documentation	The "clear" way in which you presented the requirements and the analysis, and the conclusions on the work you've done
90%	Analysis and Test	The correct analysis of the system (under the LAB3 SEAT requirements)

**Table 3: Grading**

Under the above policies, you'll also be evaluated using common sense:

- Your files will be compiled and checked; the system must work.
- Your design and architecture must be intelligent, minimal, effective, and well organized.