MIPS CPU Architecture Single-Cycle Datapath based

Third Laboratory assignment

Hanan Ribo

26.05.21

Table of contents

1. Aim of the Assignment

- Design, synthesis and analysis of a simple MIPS compatible CPU with Memory Mapped I/O.
- Understanding of CPU vs. MCU concept.
- Understanding in FPGA memory structure.

2. Definition and prior knowledge

The aim of this laboratory is to design a simple MIPS compatible CPU. The CPU will use a Single Cycle MIPS architecture and must be capable of performing instructions from MIPS instruction set. The design will be executed on the Altera Board. The MIPS architecture is Harvard architecture in order to increase throughput and simplify the logic. **You are required to support the Assembly Instruction Set given in the dedicated attached file**. For additional information regarding MIPS CPU, Architecture, ISA and instructions see MIPS technical documents [1].

3. Assignment definition

The architecture must include a MIPS ISA compatible CPU with data and program memory Caches for hosting data and code. The block diagram of the architecture is given in Figure 1. The CPU will have a standard MIPS register file. The top level and the MIPS core must be structural. The design must be compiled and loaded to the Altera board for testing. A single clock (CLK) should be used in the design.

Note: use push-button KEY0 as a System RESET (brings PC to the first program command).

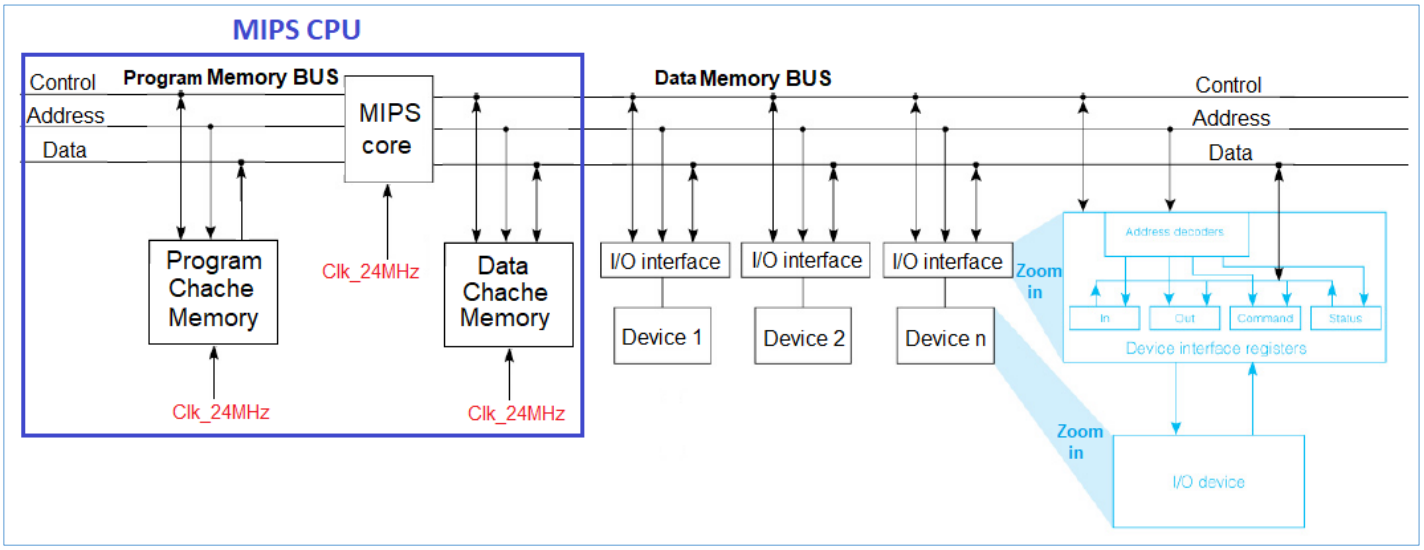
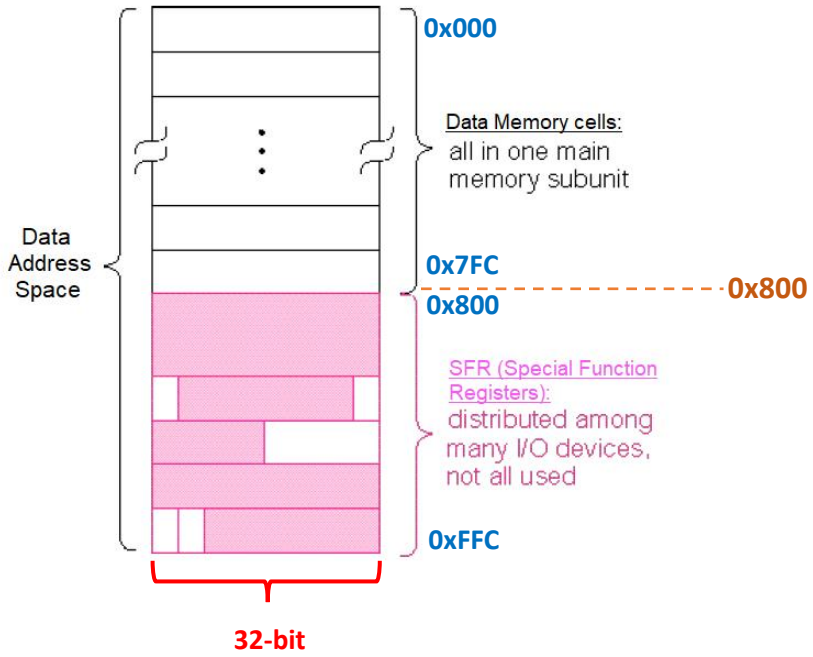


Figure 1 : System architecture

- The GPIO (General Purpose I/O) is a simple decoder with buffer registers mapped to data address (Higher than data memory) as given in the assembly code example that enables the CPU to output data to LEDs and 7-Segment and to read the Switches state.



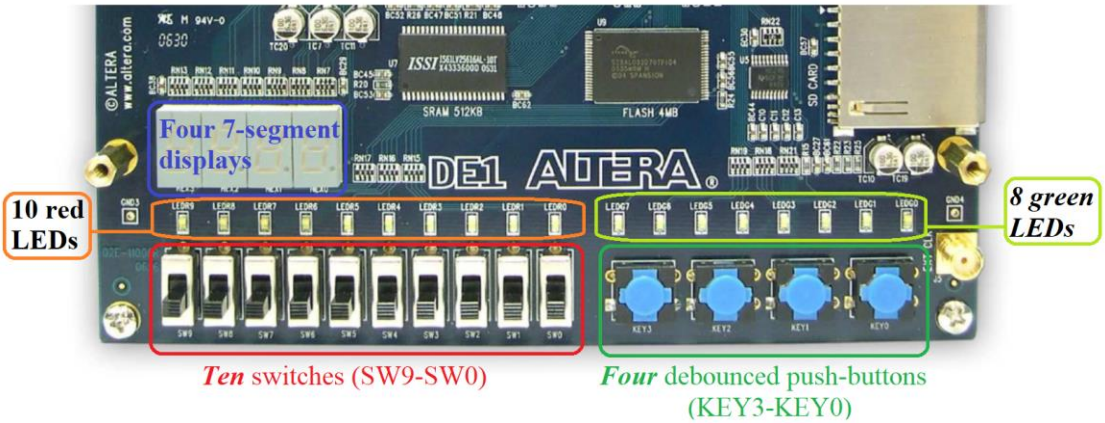
The Data Address Space is 32-bit WORD aligned where the address word is $\mathbf{0} \ldots \mathbf{0}\boldsymbol{A_{11}} \ldots \boldsymbol{A_0}$ with partial mapping.

Figure 2: Data Address Space contains Data Memory and Memory Mapped I/O

I/O devices connected:

In the hardware test case you will have to use at the FPGA board.

- Board ***ten*** switches (SW9-SW0) and push ***four*** debounced push-buttons (KEY3-KEY0) will be used as ***Input interface***.
- Board ***ten*** red LEDs (LEDR9-LEDR0), ***eight*** green LEDs (LEDG7-LEDG0) and ***four*** 7-segment displays used as ***Output interface***.




```
#-------------------- MEMORY Mapped I/O ----------------
#define PORT_LEDG[7-0] 0x800 - LSB byte (Output Mode)
#define PORT_LEDR[7-0] 0x804 - LSB byte (Output Mode)
#define PORT_HEX0[7-0] 0x808 - LSB byte (Output Mode)
#define PORT_HEX1[7-0] 0x80C - LSB byte (Output Mode)
#define PORT_HEX2[7-0] 0x810 - LSB byte (Output Mode)
#define PORT_HEX3[7-0] 0x814 - LSB byte (Output Mode)
#define PORT_SW[7-0]   0x818 - LSB byte (Input Mode)
#-------------------------------------------------------
```

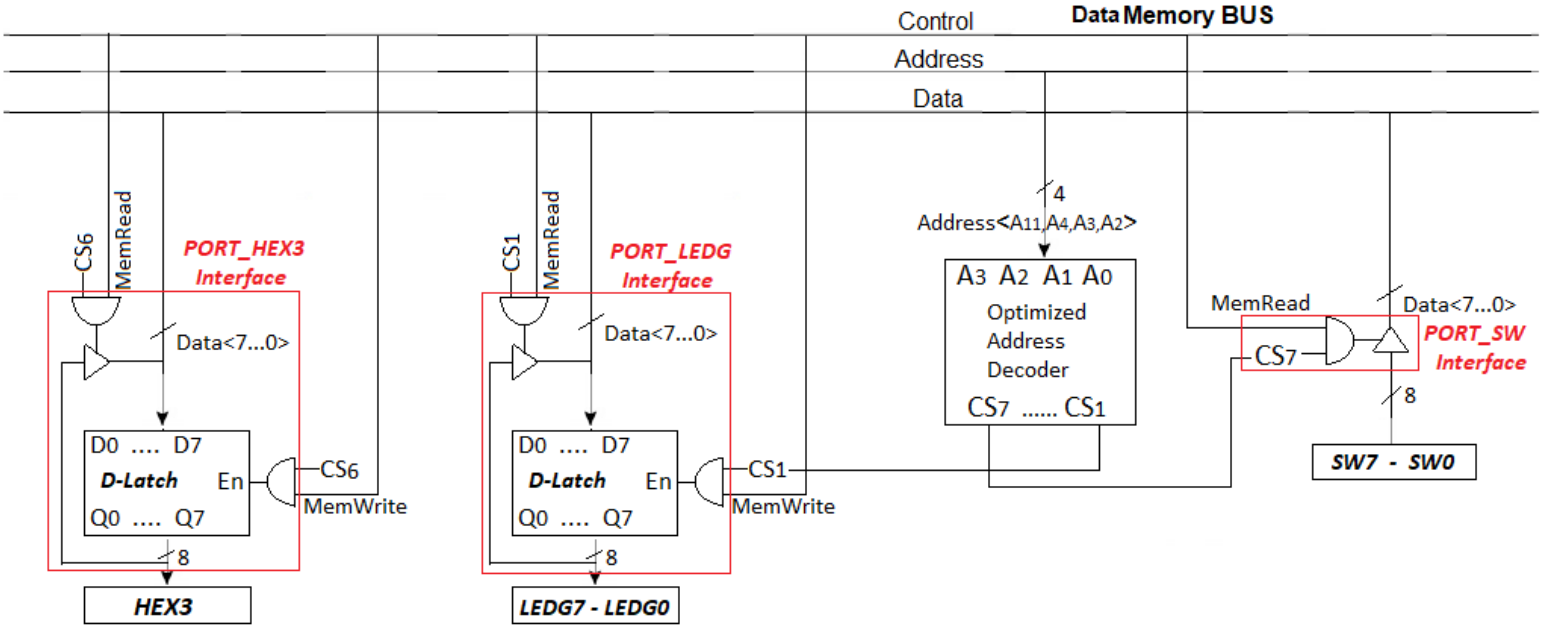


Figure 3: <u>Primitive</u> GPIO peripheral connection using Memory Mapped I/O approach

- The CPU will be based the *standard 32bit MIPS ISA* and the Instructions will be 32 bit wide. The following table shows the MIPS instruction format. For more information, see MIPS technical documents [1].

Type	**-31-**			**format (bits)**		**-0-**
R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
I	opcode (6)	rs (5)	rt (5)	immediate (16)		
J	opcode (6)	address (26)				

Table 1 : MIPS Instruction format

The Data address space is 4kB. Memory latency will be according to Table 2

Memory	**Write Latency**	**Read Latency**
Program Memory (I-Cache)	1 clk	1 clk
Data Memory (D-Cache)	1 clk	1 clk

Table 2 : Memory sizes and latency

4. Compiler, Simulator and Memory

The MARS compiler and simulator, or any other can be used to compile and simulate the assembly code. MARS compiler can also export the memory contents into the file in format that VHDL can easily read. It can also simulate a cache performance.

The mars compiler, installation instructions and documentation are available at: http://courses.missouristate.edu/KenVollmar/MARS/

5. CPU and MCU Test

a) As a test bench, you need to write assembly code, which compiles the next C code.

Note: see on Moodle file contains the Required Support of Assembly Instruction set.

```
void sort (int arr[],int sorted_arr[],int size){define it yourself …}


void main(){
     int ID={4,3,8,2,7,2,9,3},sortedID[8],IDsize=8; //int=32bit
     sort(ID,sortedID,IDsize); // IDsize=8
     while(1){
       for(i=0 ; SW0=1 ; i++) show sortedID[i&0x7] on HEX0;
      delay of 1 sec;
     }
}
```

b) Supporting of the given ***test0.asm – test3.asm*** source files.

6. Requirements

You have to do the following tasks:

- ModelSim Simulation with maximal coverage.
- Analyze the critical path, explain where it is in your VHDL design and find the maximal operating clock.
- Load the design onto the FPGA and verify the simulation results.
- **Run the required assembly source codes and explore them.**

The following must be presented in **Lab5.pdf** report file.

1. Top level block review diagram of your design.
2. For each block in the top level design:
   - RTL Viewer results
   - Logic usage for each block (Combinational and Flip-Flops).
   - Graphical description (a square with ports going in and out).
   - Port Table (direction, size, functionality).
   - Short description.
3. Maximum (Critical) path of your design – explain where it is in the code and how it is possible to optimize if you would have more time. What is the maximum clock frequency?
4. Minimum path analysis.
5. Documentation Style - Content with page numbers, Images and tables will be numbered. The caption of an images and tables below the images or tables.
6. Elaborated analysis and wave forms:
   - Maximal Frequency and critical paths from Timing Analyzer.
   - Proof of work using Signal Tap shot screens.
     Recall that, proof of work using Signal Tap is mandatory.
   - **One** basic waveform to explain the system timing.

Design requirements:

1. The design must be well commented.
2. The system must work from only one clock.
3. System RESET (KEY0) must be synchronous.
4. Conclusions

7. 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 rules violation disqualifies the task submission).
8. The **ZIP** file will contain the next six subdirectories (***only the exact next sub folders***):

Directory	**Contains**	**Comments**
VHDL	Project VHDL files	Only VHDL files, excluding test bench **Note: your project files must be well compiled (in ModelSim and Quartus separately) without errors as a basic condition before submission**
TB	VHDL files that are used for test bench	Only one **tb.vhd** for the overall DUT
SIM	ModelSim DO files	Only for **tb.vhd** of the overall DUT
DOC	Project documentation	**Readme.txt** and **Lab5.pdf** full report file
Quartus	• Signal Tap files used in project verification • Project SOF file • Project SDC file	Do not place files that are not relevant for compilation or is a result of compilation!
CODE	The assembly source code of clause 5a	

Table 3 : Directory Structure

7. Grading policy

Weight	Task	Description
20%	Documentation	The "clear" way in which you presented the requirements and the analysis and conclusions on the work you've done
80%	Analysis and Test	The correct analysis of the system (under the requirements)

Table 4: Grading

Late submissions will be not gotten.

8. References

[1]. MIPS32® Architecture for Programmers Volume I to III (from Moodle under Final Project)
[2]. ALTPLL User Guide: http://www.altera.com/literature/ug/ug_altpll.pdf
[3]. Altera RAM user guide: http://www.altera.com/literature/ug/ug_ram_rom.pdf
[4]. Altera MegaFunction User Guide: www.altera.com/literature/ug/ug_intro_to_megafunctions.pdf
[5]. Bin2Hex utility
32bit - http://www.keil.com/download/docs/113.asp
64bit - http://www.ht-lab.com/freeutils/bin2hex/bin2hex.html