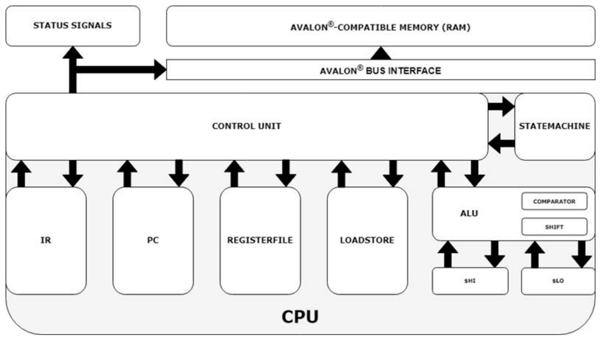
MIPS DATA SHEET - Team 3

1 CPU Architecture

1.1 A Diagram of our CPU Architecture



1.2 Architecture Overview

Block Descriptions	
MIPS CPU Bus	This block is the main interface between the RAM and the rest of the CPU. It adheres to Avalon(R) specifications.
Control Unit	This block handles the timings of the CPU and transfers data to and from the other blocks. This block is the only block to directly interface with the CPU bus.
State Machine	This block controls which state the current CPU is in - this block also enables the ability to stall, halt, and reset the CPU.
Instruction Register	This block is responsible for decoding the instruction and sending information to the Control Unit about the current Instruction Word.
Program Counter	This block keeps track of the address in the RAM of which instruction is currently being executed; setting this to 0 causes the CPU to halt.
Arithmetic Logic Unit	This block is responsible for all the arithmetic capabilities of the CPU, as well as comparator operations required for conditional branches.
Load and Store	0 / This block allows writing to the RAM as well as storing data from the RAM into the registers.
Register File	This block contains all 32 MIPS registers, and has the ability to read from two registers at any point as well as writing data to any one register.

2 Designing the CPU

2.1 Specification

2.1.1 CPU Interface:

This CPU uses a memory bus based interface, to be compatible with industrial IP blocks. Instructions and data are fetched over the same interface, and the memory has variable latency.

2.1.2 Avalon compatible RAM:

The CPU has an Avalon compatible memory-mapped interface. The CPU is independent of the memory itself.

2.1.3 Avalon Protocol:

Avalon is a clock synchronous protocol, so 'waitrequest' indicates a stall cycle, and 'readdata' is available after the read request cycle.

2.1.4 Functionality VS. Performance:

This CPU focus is stressed primarily on functionality.

2.1.5 Reset Behaviour:

This CPU will not initiate any memory transactions while the reset signal is high, as the memory may also be resetting.

2.1.6 Halt Behaviour:

This CPU is able to halt when the active signal is held at high.

2.2 Features and Advantages

2.2.1 Timing Complications

In order to avoid complications due to the precise timings of the CPU, we decided to break down the CPU into several different units.

2.2.2 Control Unit

To avoid the issue of coordinating timings between all the blocks, we decided to offload this task to the Control Unit, which sends signals to each block when the currently-executing instruction requires it.

2.2.3 Block Operation

Each block does not need to worry about timings or which cycle the CPU is in, but will simply operate on the data when and if it receives the necessary data. The separate blocks are listed in the architecture overview.

2.2.4 Standard Execution Cycle

We decided to implement the standard 3-cycle execution cycle per instruction, FETCH, EXEC1 and EXEC2, with the ability to stall each of these states and a halt state. This 3-stage execution is only possible if all the arithmetic instructions could be done in a single cycle, which we found they were.

3 Testing the CPU

3.1 Testing Approach

The process of testing and subsequent design improvement was a core part of our MIPS CPU development. Our process consisted of two main stages of testing: Initial testing of the individual CPU blocks and testing of the MIPS instruction set on the compiled CPU.

3.2 Stage 1

After each block of the CPU was initially developed, a corresponding Verilog test bench would be created alongside it to confirm that the individual blocks are implemented correctly. This step is carried out so that simple errors are dealt with early in the design process, and don't need to be dealt with later once the whole CPU is tested. Furthermore, each test bench and module were written by different members of the team to ensure that mistakes weren't carried over between them. The whole CPU is then compiled together. A further Verilog test bench must be used to test if the inputs and outputs are correct and evaluate whether it is ready to be fully tested in stage 2.

3.3 Stage 2

The instruction testing stage aims to test the whole CPU's effectiveness in executing each instruction from the MIPS instruction set. Test cases for each instruction were manually written in assembly language by team members and converted by a MIPS assembler into machine code. The CPU is compiled with the test cases loaded into its RAM. The simulated CPU output is then compared with a reference output to check if the instruction was executed correctly. If tests fail, adequate adjustments are made to the implementation as a debugging step and this complete testing process is recursively repeated for every test case.

3.4 Test cases

Test cases are split into 4 stages of testing which will be carried out in a specific order. This is important for debugging as it helps to highlight which specific blocks may be liable for errors since similar instructions will rely on the same CPU blocks. Any errors in the earlier stage instructions are likely to lead to errors in latter stage testing so it is important to follow a structured order such as this.

3.5 Stages of Test Cases

3.5.1 0 - Fundamental Testing:

Fundamental instructions, these are core to testing the other instructions which may be dependent on them and hence they are tested first.

3.5.2 1 – Arithmetic Testing:

Arithmetic instructions which all use the ALU. Most of the remaining testcases are heavily dependent on some of these instructions (mainly ADDIU and ADDU) in their testing and will require them to function.

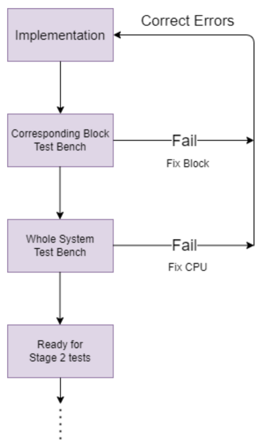
3.5.3 2 – Conditionals:

Conditional or branch instructions are tested next as they are reliant on the arithmetic instructions to produce the required conditions.

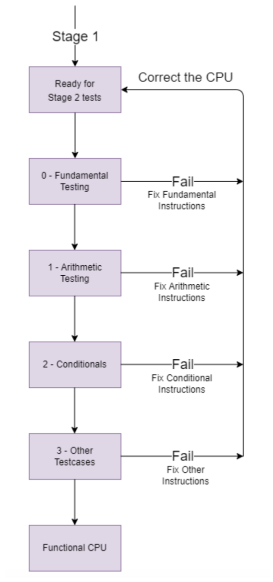
3.5.4 3 – Other Test Cases:

Finally, the remaining test cases should be tested to make sure every instruction is working. These may be slightly more complex or are dependent on previously tested instructions.

3.6 Test Flow Diagram



(a) Stage 1 Testing



(b) Stage 2 Testing

4 Area and Timing Summary

A timing and area analysis, using Intel Quartus Prime, was run by configuring the CPU onto an Intel Cyclone IV FPGA.

Area Summary	
Revision Name	`mips_cpu_bus`
Top-Level Entity Name	`mips_cpu_bus`
Total Logic Elements Name	6178/10,320 (60%)
Total Registers	1036
Total Pins	138/180 (77%)
Total Virtual Pins	0
Total Memory Bits	0 / 423,936 (0%)
Embedded Multiplier 9-bit elements	8/46 (17%)
Total PLLs	0/2 (0%)

Timing Summary		
FMax	Restricted FMax	Clock Name
4.61 MHz	4.61 MHz	ControlUnit:cpuCU|| IW[29]
64.36 MHz	64.36 MHz	ControlUnit:cpuCU|| statemac:cpuSM||e1
143.04 MHz	143.04 MHz	clk

5 References

Price, C., 1995, *MIPS IV Instruction Set* [Online]
Available at: https://www.cs.cmu.edu/afs/cs/academic/class/15740-f97/public/doc/mips-isa.pdf [Accessed 15 December 2021]
Soares, J., and Rocha, R., 2019-2020, *Encoding MIPS Instructions* [Online]
Available at: https://www.dcc.fc.up.pt/~ricroc/aulas/1920/ac/apontamentos/P04_encoding_mips_instructions.pdf [Accessed 16 December 2021]
OpenCores, 2019-2020, *Most MIPS I(TM) Opcodes* [Online]
Available at: https://opencores.org/projects/plasma/opcodes [Accessed 16 December 2021]
Berkeley, *MIPS Reference Sheet* [Online]
Available at: https://inst.eecs.berkeley.edu/~cs61c/resources/MIPS_help.html [Accessed 17 December 2021]
Pakin, S., 2021, *The Comprehensive LATEX Symbol List* [Online]
Available at: http://tug.ctan.org/info/symbols/comprehensive/symbols-a4.pdf [Accessed 17 December 2021]