AM-13 MIPS CPU

The AM-13 MIPS CPU is a MIPS1 ISA compliant CPU core that is optimized for embedded systems. Its lightweight design makes it perfectly suited for applications with stringent power requirements and where chip real estate comes at a premium. A test-driven development methodology used at all stages of the design process robustly guarantees functional performance even under the most demanding of applications. The CPU comes ready to go out-of-the-box with an industry standard Avalon memory-mapped bus interface that simplifies its integration with peripherals, thereby reducing development costs and time-to-market.

Features	
32-bit MIPS1	8.12 MHz max clock speed
Registers	32 general purpose registers
	Register 0 fixed at value 0
	31 user-writable registers
	Dedicated HI and LO registers for multiplication and division operations
Single-Cycle Execution	All instructions are executed with a CPI (cycles per instruction) of 1 (excluding memory fetch cycles)
CPU State Flag	Active Flag indicates when CPU has completed program execution
Specialised Modules	'lwlr' module for LWL and LWR instructions
	'ALU Control' module that determines control signals for the ALU
	'ALU' module not only performs arithmetic operations but also determines whether branches are taken.
	'Sign Extend' module performs the necessary sign extending operation before the data can be written into a register or enter the ALU
	'Instruction Register' stores the instruction data from the Instruction Memory
	'PC Next', a fully combinatorial module that calculates the next PC value
Avalon Bus Controller	Avalon-MM Interface compliant bus controller eases integration with peripherals
	Compatible with variable latency slaves

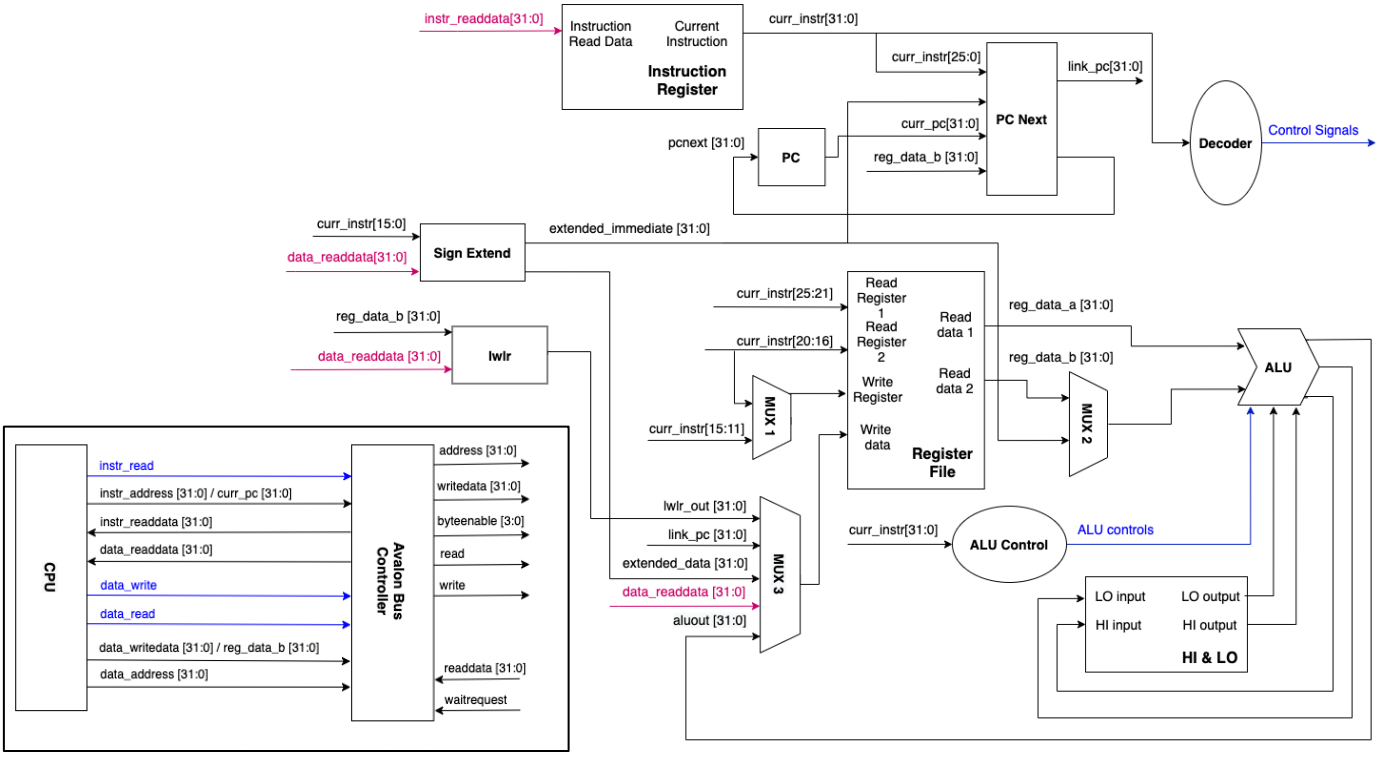

Figure 1. CPU Block Diagram

Figure 2 (Bottom Left): CPU and Bus Controller Interface

Design Decisions	
Modularizing large components such as the Decoder and ALU	Modularisation allows for efficient testing of each component and subsequently the integration of all these components to obtain the final product.
	Guarantees reliability in terms of component functionality as each component undergoes rigorous tests before it is utilised.
Separate control unit for decoder and ALU	Separating control units for the ALU and the rest of the CPU improves the efficiency of the CPU decoding.
	Further reduces the possibility of error as the size of the decoder module is reduced, allowing for more efficient testing
	Allows the ALU and its control unit to be tested without requiring the rest of the CPU, which ensures the functionality of the ALU
Specialized module to sign-extend necessary data	Having a stand-alone module for this purpose simplifies data manipulation as it passes through the CPU, ensuring that the bit width of data entering and exiting modules are always according to specifications.
Unique 'lwlr' module that performs LWL and LWR instructions	Due to the complicated nature of these instructions, a separate module that handles this operation makes testing of the CPU more modular and less error prone.
Bus controller module	Simplicity when converting from a Harvard to a Bus interface
	Choosing whether to use this controller module allows the same CPU to interface with both a Harvard and Bus based memory interface.
	Facilitates isolated testing and verification of the bus controller separately from the CPU which helps narrow down fault locations.

Testbench

The testbench is used to ensure the functionality and correctness of the MIPS CPU by detecting any potential logic flaws and design deficiency based on an extensive test case set. The two main tasks of the testbench are, firstly, generating testcases for each instruction, and secondly, creating scripts which integrates the process of compiling, generating executable files, capturing return codes and outputs, and finally comparing the result.

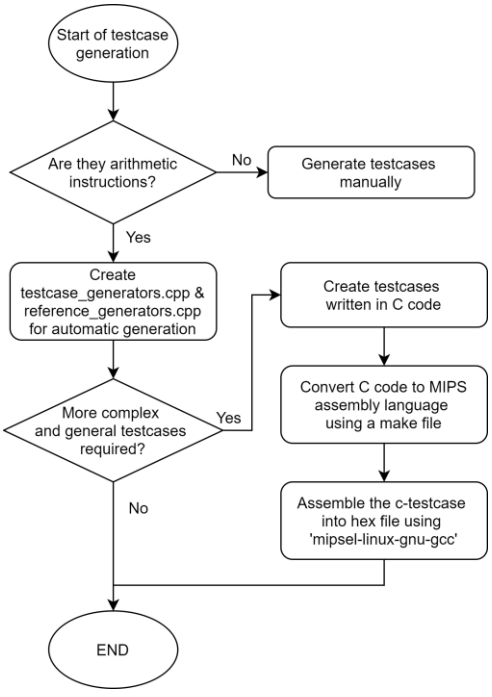

Figure 3. Testcase Generation Flow Chart

The key process of the testcase generation process is shown in Figure 3.

- Both automatic and manual generation methods are used for test cases and reference generation.

- Automatic generation: Values for arithmetic instructions are separated by ranges to ensure all possibilities are tested, and the rand() function used guarantees complete randomness.
- Manual generation: Handwritten test cases lead to more precise edge case detections.

- Testcases with more complex logic are created using a Makefile that converts C code to MIPS assembly language.
- Most testcases are comprised of only the five fundamental instructions and the instruction in question. This allows the testbench to accurately single out the exact instruction that has failed.
- Comments are added to test cases and would be output with the result of the test case, which describes the testing aspect of the current case making the testing process more intuitive.
- At least five test cases are provided for each instruction to test CPU comprehensively.

Additionally, the testing flow and the critical testing approaches of testbench are shown as Figure 4 below.

- The testbench runs by running the test script with the {source_directory} argument and an optional {instruction} argument
    - If the instruction argument is specified, each test case for that instruction is tested
    - Else, all test cases for every instruction would be tested
- When running all test cases, five fundamental instructions – LUI, ADDIU, LW, SW and JR would be tested first. If any of them fails, the rest are likely to fail as well.
- Assembler checks the validity of instructions and rejects any test case with a wrong instruction format. It groups instructions together for efficient assembling and locates which line any error within testcases could be, to allow for easier test case debugging.
- Various error messages are provided when failing at different stages. It helps to locate the error and allows further modifications of the CPU correspondingly.

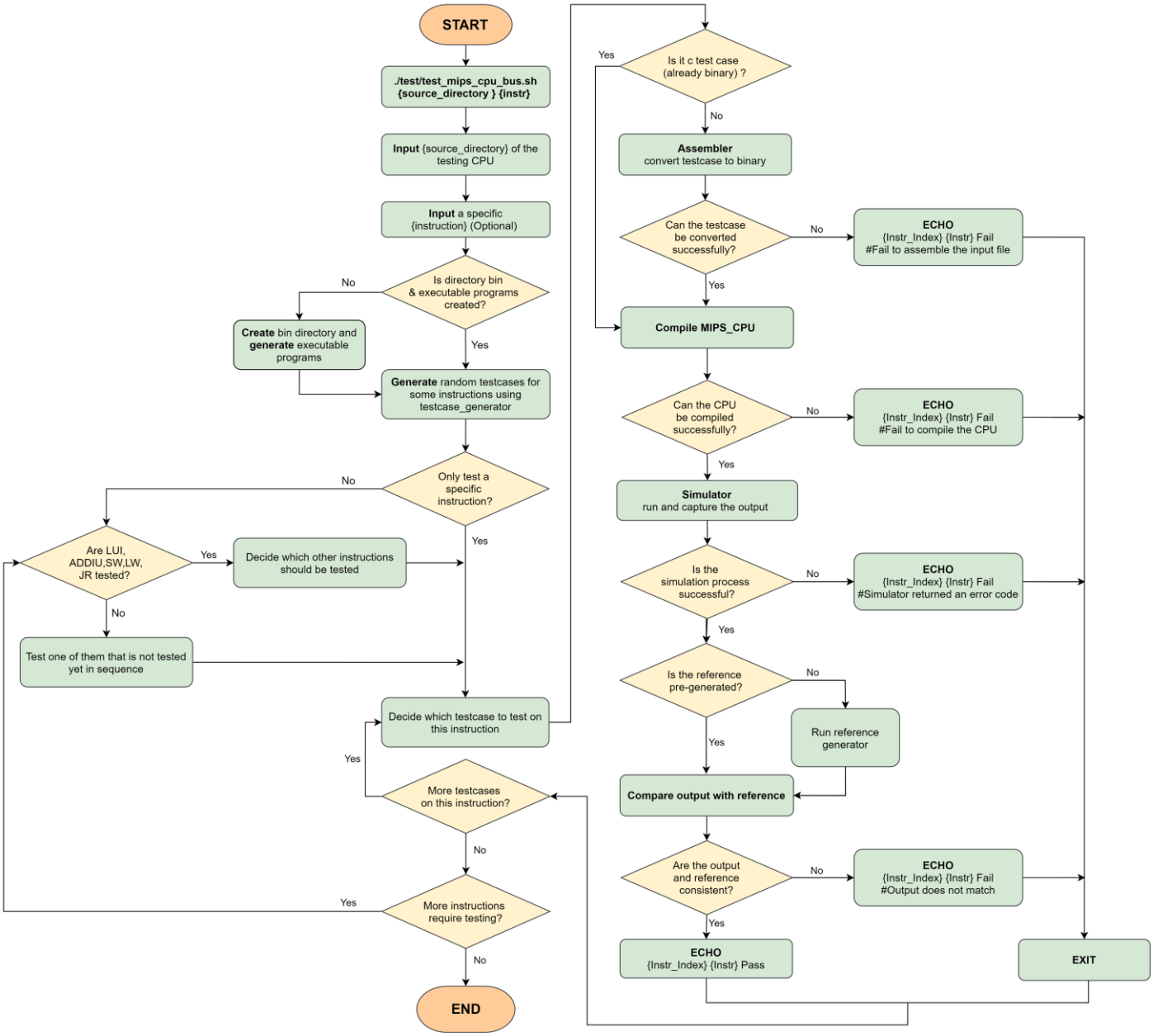

Figure 4. Testing Process Flow Chart

Area and Timing Summary

The AM-13 MIPS CPU's max clock frequency of 8.12MHz is suited for low-power and low-cost applications where higher clock frequencies are not required by other peripherals. Its single-cycle instruction execution potentially simplifies operation coordination with other peripherals.

	Resource	Usage
1	Total logic elements	9,083 / 15,408 (59 %)
1	-- Combinational with no register	7793
2	-- Register only	454
3	-- Combinational with a register	836
2		
3	Logic element usage by number of LUT inputs	
1	-- 4 input functions	4985
2	-- 3 input functions	3163
3	-- <=2 input functions	481
4	-- Register only	454
4		
5	Logic elements by mode	
1	-- normal mode	6012
2	-- arithmetic mode	2617
6		
7	Total registers*	1,290 / 17,056 (8 %)
1	-- Dedicated logic registers	1,290 / 15,408 (8 %)
2	-- I/O registers	0 / 1,648 (0 %)

Figure 5. Area Summary

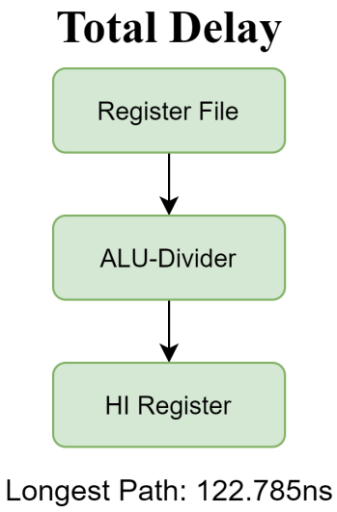

Figure 6. Critical Path

To achieve single-cycle performance, combinational logic dominates the AM-13 MIPS CPU's area usage with 7793 logic elements. The largest module is the combinatorial ALU, which implements all four basic arithmetic operations: addition, subtraction, multiplication, and division. Multiplication and Division each have their own fully combinatorial path which contributes greatly to area usage.

The combinatorial paths in the ALU module are a limiting factor for the CPU's maximum clock frequency. The critical path in the CPU runs from the register file, through the ALU Divider and into the HI register as shown in Figure 6. This data path has a delay of 122.785ns[1]. The lengthy delay is caused by data having to traverse the combinatorial gate path in a full 32-bit divider, the interconnect that links them together and to the source and destination registers.

The centralised decoder module also has extensive combinatorial logic paths as it generates the control signals for all other modules based on the instruction word input.

The CPU has a nominal register usage mainly from the register file. The Avalon bus controller also utilises several registers to hold data for internal states and the CPU.

[1] Analysed with Quartus Prime Lite 19.1.0 on a Cyclone IV E EP4CE15F23C6