JoSDC'24

المسابقة الوطنية لتصميم الشرائح الإلكترونية

Jordan National Semiconductors Design Competition

Qualifying Phase Guidelines and Reference Design JoSDC'24

1. Introduction: Phase description and requirements

Phase Duration: 22/9/2024 – 13/10/2024

In this phase, teams will prepare for the competition by demonstrating their readiness to proceed further in the upcoming stages of the competition. **The theme of this phase is verification**, as teams will focus on validating and debugging the baseline design provided. You must pass this qualifying phase to continue your journey in the competition.

2. Problem Statement:

Teams are expected to validate, debug, and fix errors in the MIPS-based baseline design. The baseline design is a 32-bit single-cycle processor that supports the following subset of instructions:

add	addi	sub	and	or	slt	lw	sw	beq

The implementation of the baseline processors in Verilog will be provided. However, the code includes unknown number of intentionally injected bugs. Teams are responsible for testing the design, tracing the code, identifying and correcting syntax, logical, and runtime errors, and ensuring the processor functions as intended.

No big changes to the overall design are required—only error correction. Additionally, teams must document all identified errors (see bug report appendix), and provide supporting information such as signal analysis and performance data, using the provided template.

2. Datapath

The MIPS baseline datapath, which will be the focus during This Phase, is depicted in Figure 1. This datapath is fully sufficient for handling the 9 instructions mentioned earlier *(add, addi, sub, and, or, slt, lw, sw, beq)*. Therefore, no additional units or modifications are required at this stage. Your task is to study the given design thoroughly, and to validate its functionality and debug any defects.

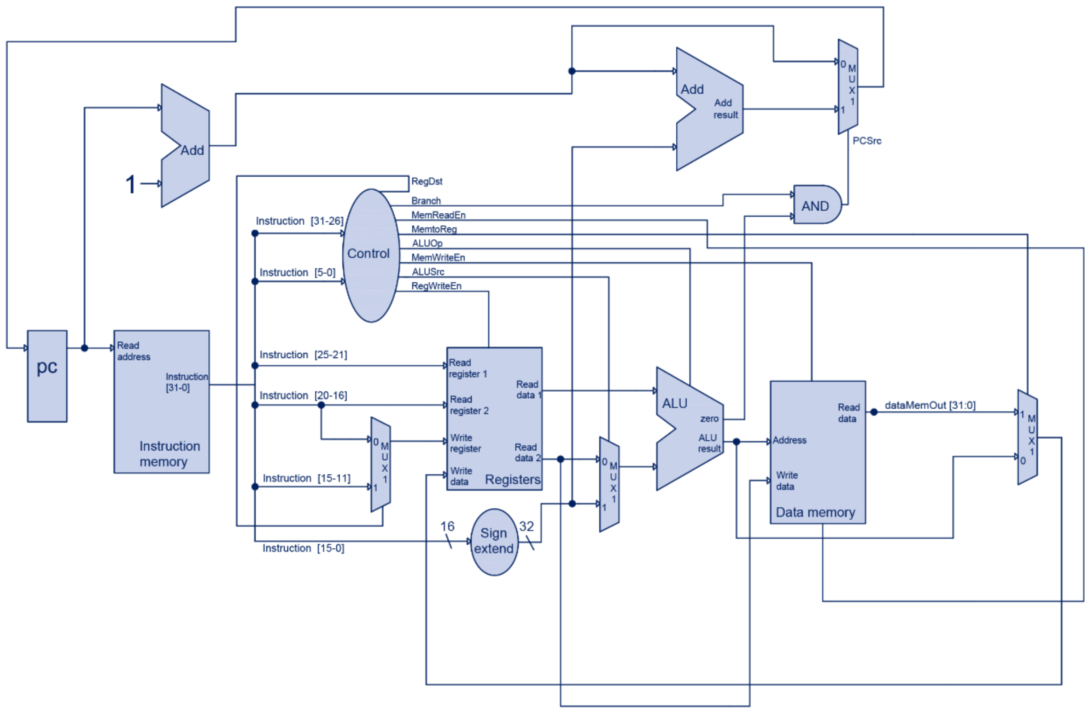

Figure 1 Baseline Datapath

3. Control Unit Signals

Control unit is the "brain" of the CPU architecture. All teams are required to thoroughly analyze and trace the execution flow within the design. Therefore, it is essential to fully understand the control unit. You must be familiar with all signals used in the baseline design, their function, and how to handle them. The signals used in the baseline design are listed below, along with more details about each:

Table 1 Control Unit Signals

Signal name	Functionality	Usage
RegDst	Selects which register is the destination (Rt or Rd).	0 -> select Rt as destination register
		1 -> select Rd as destination register
Branch	Determines if the instruction branch or no	0 -> instruction is not a branch
		1 -> instruction is a branch
MemReadEn	Enables reading from memory	0 -> prevent reading from memory
		1 -> allow reading from memory
MemtoReg	Controls if data comes from ALU result or memory to register file	0 -> select ALU Result
		1 -> select memory output
ALUOp	Determines the operation the ALU should perform	0 -> addition operation
		1 -> subtraction operation
		2 -> logical ANDing operation
		3 -> logical ORing operation
		4 -> comparing
MemWriteEn	Enables writing to memory	0 -> prevent writing on the memory
		1 -> allow writing on the memory
ReqWriteEn	Controls whether the register file is written to	0 -> prevent writing on the register file
		1 -> allow writing on the register file
ALUSrc	Selects between register operand or immediate value for ALU	0 -> selects Rt as second ALU operand
		1 -> select immediate as ALU operand

Make sure to trace each signal as part of your verification and debugging process to ensure the correct operation of the processor.

4. Conclusion

At the end of the Qualification Phase, teams are required to **submit** the following:

1. **Qualifying Phase Documentation using the provided Template**
   This PDF document should contain all necessary data, including identified errors, signal analysis, and other required information as outlined during the phase. You **must use the provided template.**

2. **Project Folder**
   This folder must include the corrected Verilog code and the complete project, free of syntax errors, ready to be compiled and run. The project should also be configured with a benchmark, prepared for testing.

Submission Method:
Submission will be through Google classroom. Details will be announced later.

Submission deadline: October 13, 2024

The results will be announced ASAP.

5. Resources

- Textbook (Computer Organization and Design for Petterson and Hennessy)
- Online Resources (online resources are there to help you, not to replace you!)
- Your University Professors
- Dr. Dheya Mostafa (Project Manager of JoSDC Competition) – through MS Teams
- JoSDC Mentors – Through MS Teams, list of names shown below

Table 2 Mentors information list

Mentor Name	Contact email (in MS Teams)	University
Abdalrahman Alshannaq	alshannaq.abd	HU
Issa Qandah	essa.qandah2	HU
Hassan TaqiEddin	hassantaqi1812	HU
Ammar Zaidan	ammar01rz	HU
Tamara Ghatashe	ghatashet	HU
Abdullah Matar	matar.abdullah03	PSUT
Yara Altamimi	yara.ashraf292	PSUT
Mahmoud Abu Qteish	mahmoud.abuqtiesh22903	PSUT
Jude Altheeb	jud20210415	PSUT
Abdalrahman Ebdah	abd0211201	JU
Omar Sweiti	amr0214000	JU
Al Baraa Al Najjar	alb0217928	JU
Dawoud asasfeh	dao0218788	JU

** Mentors are students (like you) who dedicate their time to help you. Be patient and considerate when contacting them

6. Appendix: Bug Report Manual

When finding and correcting an error in the Datapath, there are systematic process you need to follow to ensure the error is resolved thoroughly. These steps include reproducing the error, identifying the root cause, applying the fix, and verifying the solution through extensive testing. Additionally, update all relevant documentation and communicate the resolution to the team for proper tracking and validation.

In this phase, each bug must be documented using the provided **Bug Report Template**. Below is a clarification of key bug report terms to help you complete the template accurately.

Key Bug Report Terms

1. Bug Title: A short, descriptive name for the bug. The title should provide a quick overview of the issue.
2. Bug ID: A unique identifier assigned to each bug. This helps in tracking the bug through its lifecycle.
3. Bug Type: Classify the bug (e.g., "Syntax", "Logic", "Performance", etc…). This helps categorize the error for easier resolution and reporting.
4. Reported by: Name of the person or team who discovered and reported the bug.
5. Open Date: The date when the bug was reported and entered into the tracking system.
6. Assigned to: Name of the person or team responsible for fixing the bug.
7. Close Date: The date when the bug was resolved and confirmed fixed.
8. Description: A detailed description of the bug, including what it affects and any relevant context.
9. Steps to Reproduce: A list of actions that can be followed to trigger the bug.
10. Expected Behavior: What the processor should have done if no bug existed.
11. Actual Behavior: What actually happened when the bug was encountered.
12. Solution Implemented: The fix that was applied to resolve the bug.

See bug sample in next page!

Table 3 Bug Sample

Bug Title:	Incorrect ALU Operation for sub Instruction		
Bug ID :	Bug#01	Bug Type	Logic Error
Reported by:	Person X	Open Date	22/9/2024
Assigned to:	Person Y	Close date	25/9/2024
Description	When running the sub instruction, the ALU adds the operands instead of subtracting them due to an incorrect control signal being set for the ALU operation.		
Steps to reproduce	1. Run the processor with two operands using the sub instruction. 2. Observe the output in the register file. 3. The result shows addition instead of subtraction.		
Expected Behavior	The ALU should subtract the second operand from the first and store the result in the destination register.		
Actual Behavior	The ALU adds the two operands and stores the sum in the destination register.		
Solution implemented	The ALUOp signal in the control unit was incorrectly set to 00 (add). It was changed to 10 to correctly perform subtraction. Modified the control unit logic in the Verilog file at line x.		