Laboratory 7: Single-Cycle Computer Implementation EE 203 Digital Systems Design

[Kerim Ercan], [042401167]
Berk Özkan [042202019]

[Egemen Dönmez [042202024]

[Alpar Kaan Güven []

MEF University, Istanbul, Turkey

Section : [1]
Date Performed: [20.12.2024]
Submission Date: [22.12.2024]

Abstract

This report presents the results and analysis of the [23] laboratory session for the EE203 Digital Systems Design course. The emphasis is placed on the interpretation of results and their alignment with theoretical expectations. It also highlights the critical steps taken to achieve the objectives, including schematics, screenshots, and waveforms where applicable. Hardcopy reports without proper electronic submission or lacking signatures will not be accepted.

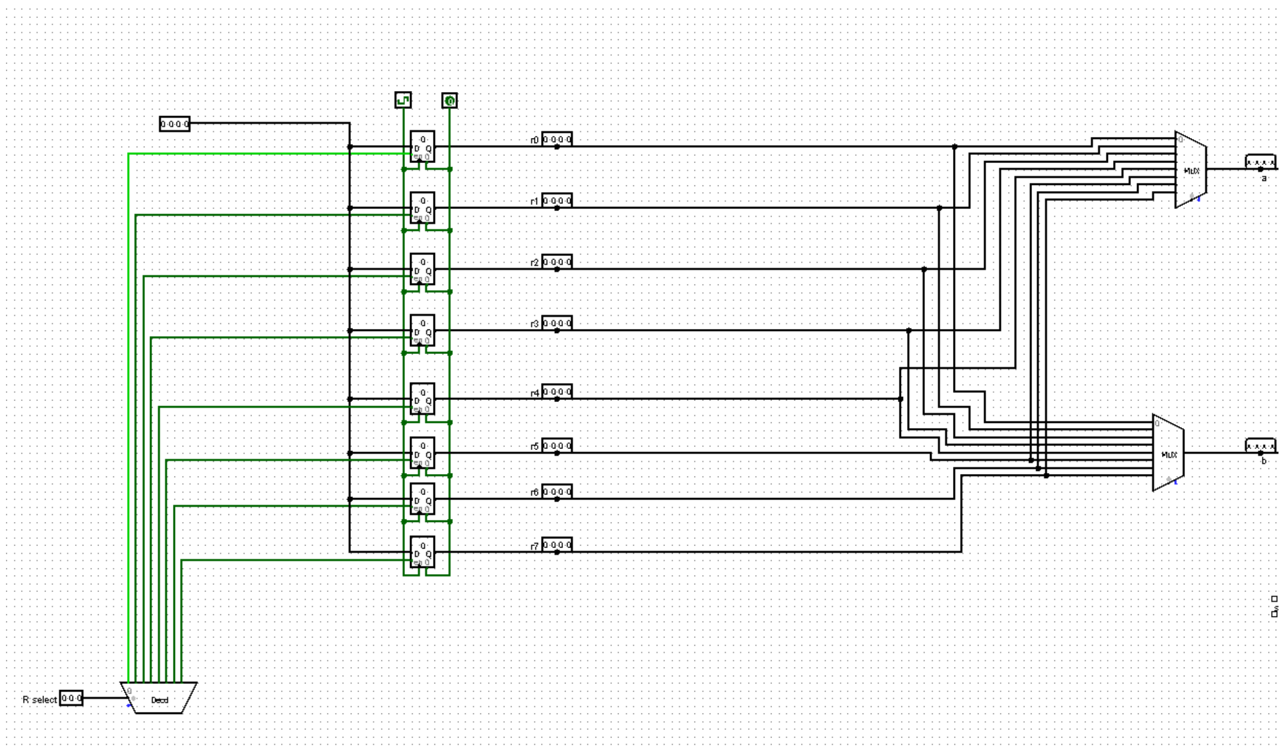
1 Introduction

Implementing the single-cycle computer's datapath was the aim of this lab session. The main goal of the session was to learn how to effectively use the registers and the ALU in a practical activity, such adding a constant to a register value and saving the outcome in a different register. Key findings from the testing process, a description of the implementation techniques, and a critical analysis of the difficulties faced and the solutions employed are all included in this paper.

2 Theory

3.1 (30 points) In this assignment, you will implement the datapath of a single-cycle computer in Logisim. To be able to connect the building blocks of the computing machine, you will have to insert ?irst

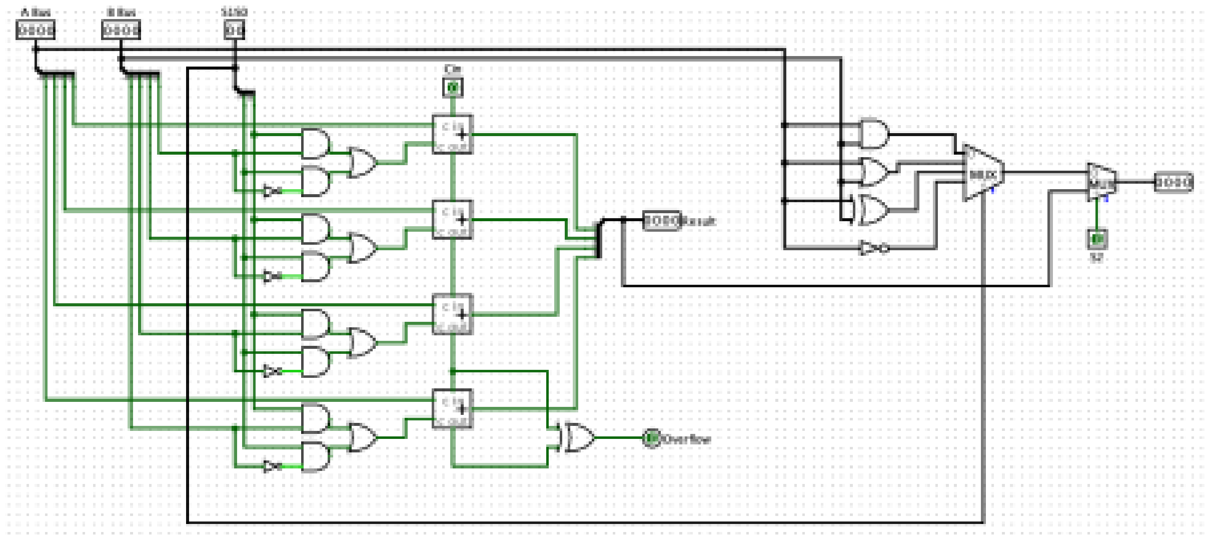
• Eight general purpose registers where registers can store 4-bits i.e., $n = 4$.



We have a decoder and eight registers. Each decoder output is used to modify values stored in a separate register and is linked to the appropriate register's enable input. A and B are chosen from the outputs of every register using the two MUXs on the right.

• An ALU capable of running both arithmetic as well as logical operations. You do not need to worry about the status bits of the ALU for this assignment.

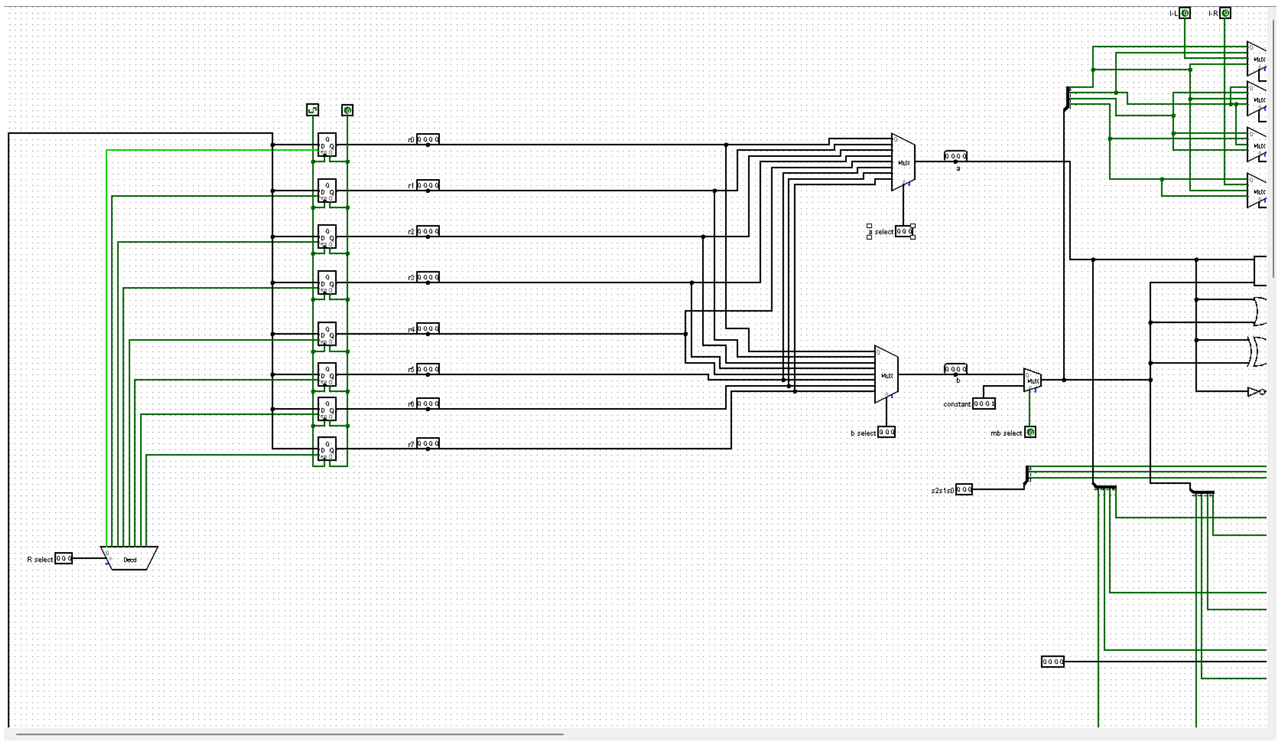
2
1

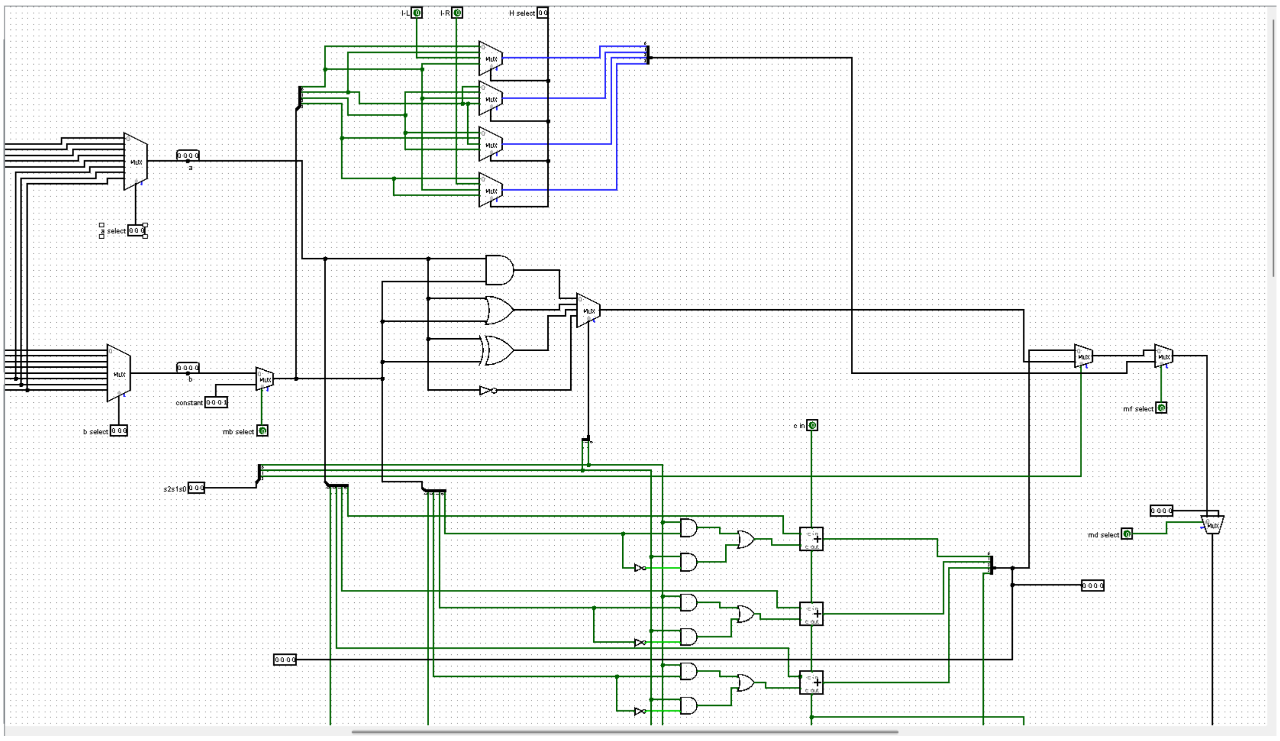


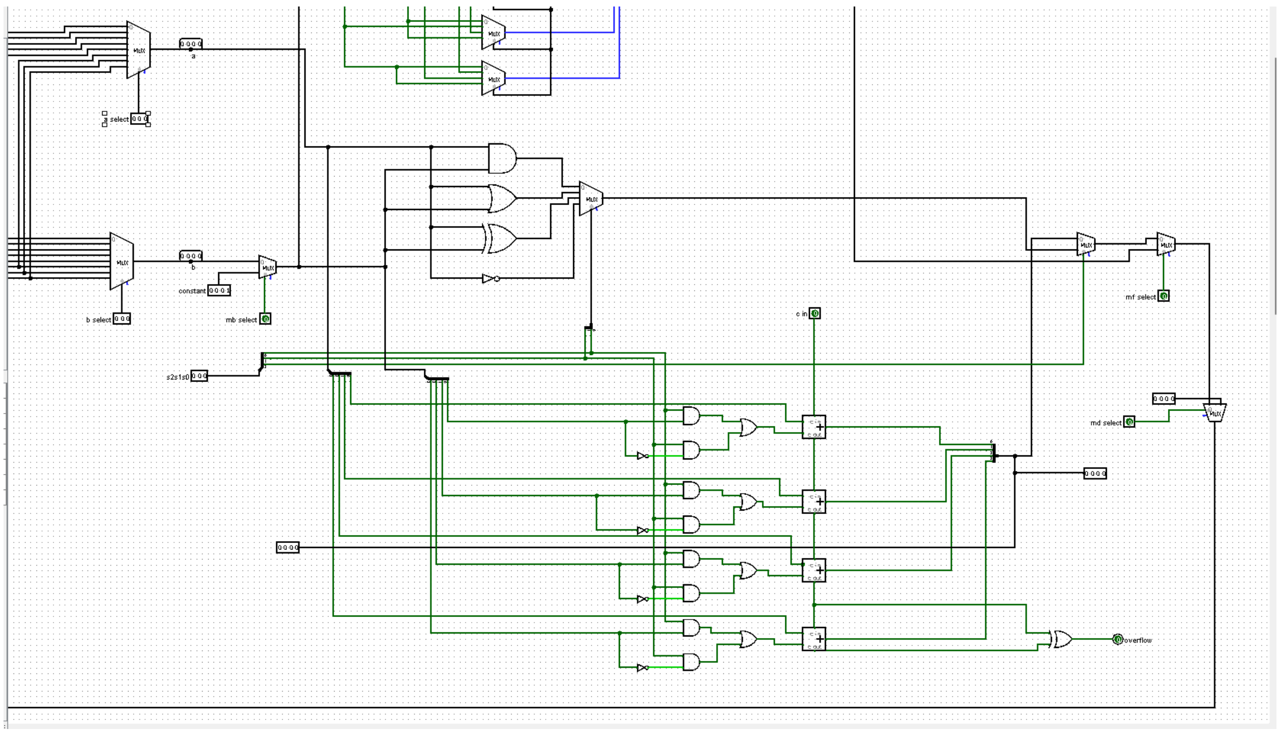
Adder has s0 and s1 select inputs for operation select and s2 is used to change between adder and logic operations. ALU also has a overflow output.

3 Experimental Results

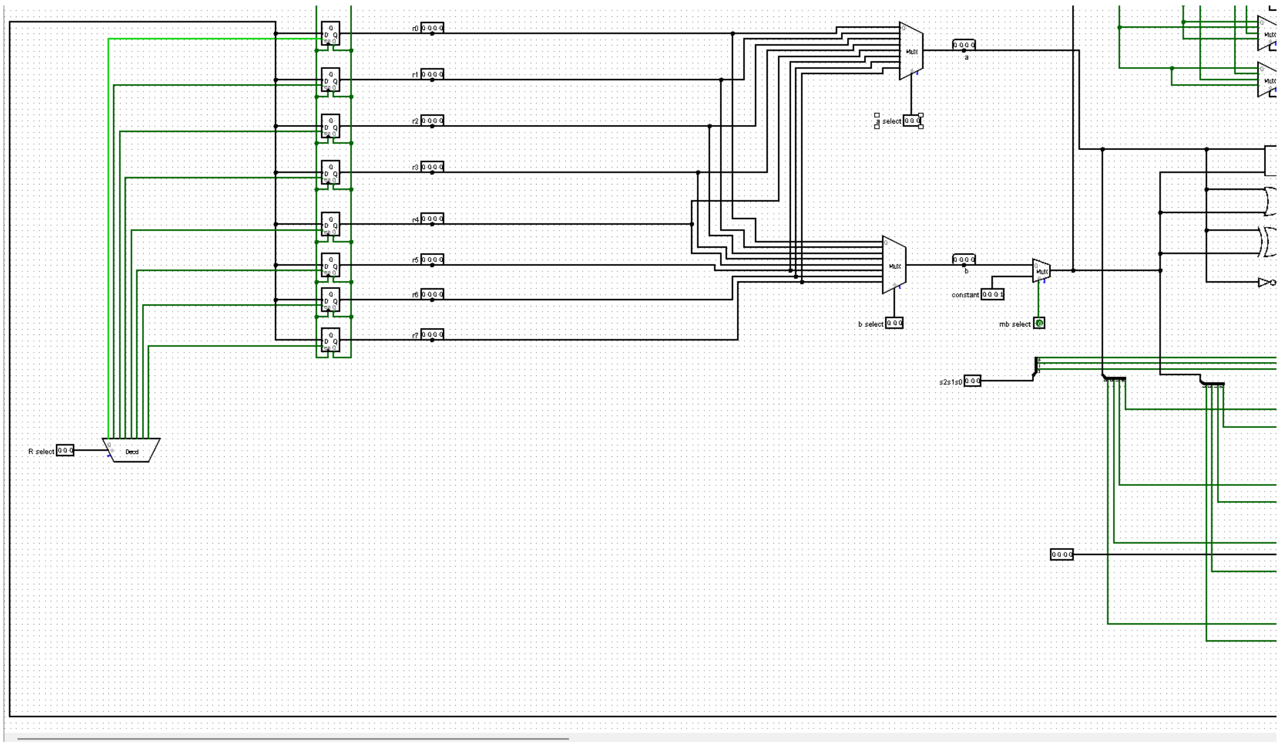
3.1



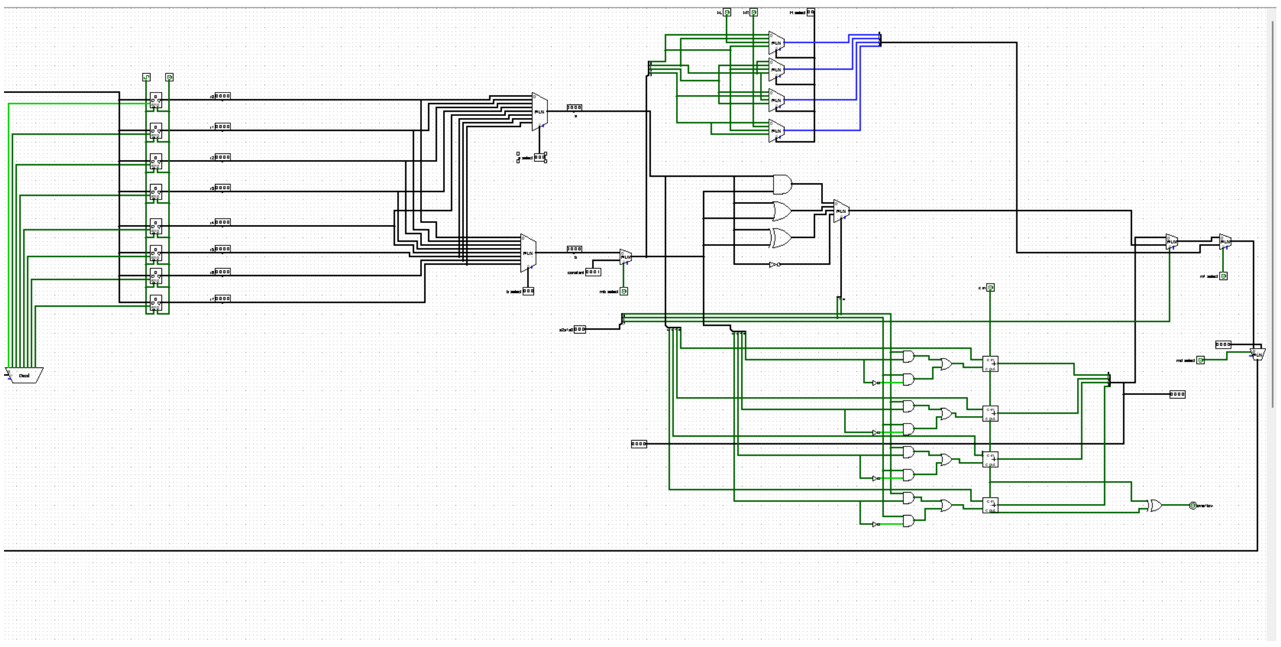




MUX
a
select
b
constant 0001
mb select
b select
s2s1s0
c in
md select
mf select
overflow



r0
r1
r2
r3
r4
r5
r6
r7
R select
Decod
MUX
a
select
b
constant 0001
b select
mb select
s2s1s0



3.2 (Bonus+10) Implement the hard-wired control logic similar to the one given in class slides. You can take away unnecessary part from this logic as your computer do not need to support branch/jump and memory operations. With this step, you should be able to see the big picture shown in Figure 2.

4 Discussion of Experiment Results

4.1 (5 points) Please discuss the results of the experiments. Generate three questions and responses. What kind of dif?iculties you faced during the simulation? Explain them and talk about the the methods to resolve.

Question 1: How is the shifter implemented?
We arrange four input bits in accordance with the desired shift operation determined by the inputs I-L and I-R by employing four MUXs with a two bit select code.

Questıon 2: How do you decide which register to modify?
You can choose from eight registers by utilizing a selection input and a decoder.

question 3: How do select inputs work?
To choose between arithmetic and logic processes S2 is utilized. To choose the arithmetic or logic operation to do, use S0 and S1 The arithmetic component of the ALU determines whether B is used as its own complement or as its own, as well as whether it is used as itself. It ıs a select input for a MUX with two select inputs in the logic section.

4.2 (5 points) What are the disadvantages of not having branch and jump operations? Discuss within the context of writing C/Java programs.

The absence of department and jump operations substantially limits a device's capacity to deal with control waft in C or Java packages. Without them, loops and conditionals like `if` or `at the same time as` cannot function without delay. This forces guide handling, making software logic extra complex and increasing code size. Such a drawback reduces flexibility and makes it tougher to resolve actual-world issues, where choice-making and new release are essential for powerful application execution.

4.3 (5 points) List the set of main disadvantages of single-cycle computers compared to multi-cycle computers. You are free to search over the web for the answers of these questions.

1.Inefficient Execution Time
In single-cycle computers each guıdance, whether or not easy or complex takes the same fixed amount of time to execute. This method uses less difficult commands like register-to-sign in operations waste cycles anticipating the clock to finish. This inefficiency results in terrible overall performance particularly in packages with mix of simple and complex instructions.

2.Increased Critical Path Delay
The clock cycle time in single-cycle computers have to be lengthy sufficient to house the slowest practise which includes reminıscence get admission to or multiplication. This creates a important route put off where faster commands are bogged down to healthy the slowest operation, reducing the general pace and performance of the system

3.Higher Power Consumption
The want for a consistently excessive clock frequency in single-cycle computer systems outcomes in higher power usage. Since all commands run at the equal pace even if pointless for simpler operations the machine consumes extra power compared to a multi-cycle layout where resources are used greater effıcaciously.

4.Scalability Challenges
As the complexıty of the education set increases, unmarried-cycle designs emerge as harder to manipulate. Adding new commands or ıncreasing functionality regularly leads to multiplied delays in execution, making it hard to keep synchronization and overall performance.

5.Inefficient Resource Utilization
Single-cycle computer systems require all hardware components to be active for each education even though some components are not needed. For instance the ALU may also continue to be energetic at some poınt of memory get admission to operations main to wasted electricity and assets.