Digital Logic Design Lab

Design Project

Mini Computational Logic Unit

Submitted by:

Tahmooras Khan	22L-7484
Huzaifa Amir	22L-7518
Ahmed Abdullah	22L-7503

Submitted to:
Mr. Salman Shoaib

10/05/2023

Department of Electrical Engineering

National University of Computer and Emerging Sciences, Lahore

Table of Contents

Introduction

The purpose of this Mini Computational Logic Unit is to develop a small unit which can control the flow of Data, process the Data and provide output. It will be working on registers as our variables, and we will be processing 3 different functions on them.

In this project, we have used decoders to control the flow of data and used the concept of enabling the data inputs to control the data. It will be taking 6 OP-CODE Binary Switches as inputs, all of them are also connected to enable to stop the flow of unnecessary data.

The advantage of this MCU will be to provide an excellent system of performing operations on the registers, saving the data, and applying operations on the same data.

The first major function of this MCU will be getting input from users, we have added 4 additional binary inputs for users to keep updating the data and perform functions.

The second major function of this MCU will be shifting the data 1 bit forward or backward within the registers. To avoid the loss of data from our array, we have inverted the pushed data back to the register itself in the form of a circle or loop.

The third major function of this MCU will be performing arithmetic functions on the registers. Since values can or cannot increase and be out of the subscript of the register we have added an overflow and carry register to tell the user that the data is overflowed or carried.

Problem Analysis

The limitations of current control systems, which are frequently created for particular input/output configurations and are not easily adaptable to new configurations, are the root causes of the issue. Inefficiency and difficulty in use are the problem's effects, which can reduce productivity and make users more irritated. Depending on their individual needs and requirements, users are impacted by the problem in different ways.

Potential remedies for this issue include the use of decoders and other electronic parts to build a more adaptable and effective control system. Decoders and other parts could be included in a Mini Controlling Unit to allow control of various input/output configurations from a single device. As a result, fewer individual controllers would be required for each device or system, increasing efficiency.

Design Requirements

1. Binary Switches
2. Binary Probes
3. Registers with Enable
4. Registers without Clear
5. Clock
6. Latches and Flip Flops
7. Decoders 3-8
8. Decoder 4-16
9. Decoder 1-2
10. MUX
11. Logic Gates
12. Vcc and Ground

Feasibility Analysis

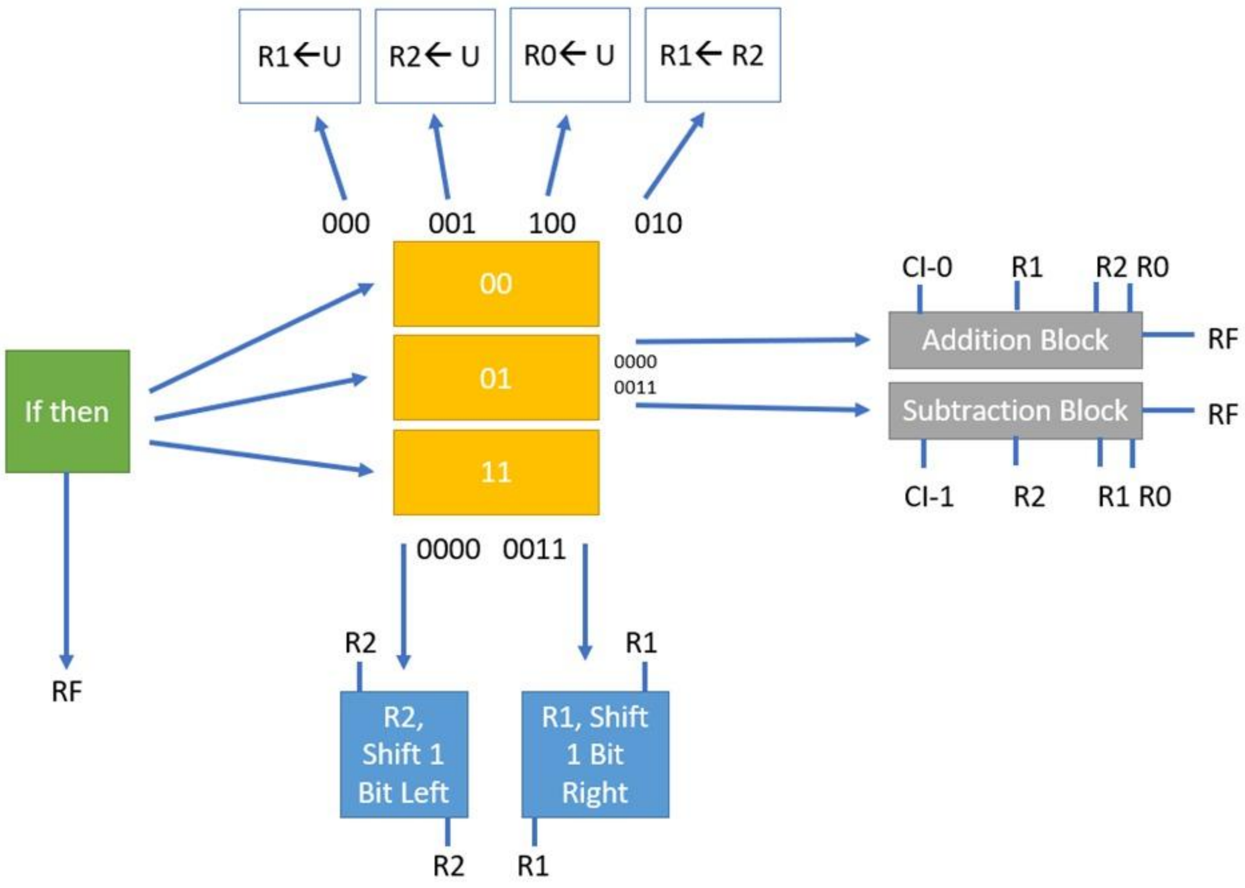
In the OP codes provided, the 3rd bit from MSB serves no real purpose. So instead of using a 4*16 decoder (to take inputs from OP 2-5) a 3*8 decoder is used. the inputs in this decoder are bits of OP 3-5. In order to utilize the 3rd bit from MSB, it’s NOT given as input in all 3*8 decoders. While performing this, the input is first given to an AND gate with another input (which essentially decides whether the decoder will be enabled). Then the output of the AND gate is given as input in enable.

Possible Solutions

In the following project, 3 3*8 decoders were used. While other approaches, such as using multiplexers, could be implemented, it was most suitable to use decoders. If multiplexers were used, a lot of them would be needed. For example, where there is one 2*4 decoder, we would be requiring 5 2*1 multiplexers to do the job. This means that implementing Muxs here would be more expensive as more ICs would be used. Similarly, we could have used Mux instead of gates while dealing with the inputs to the registers.

Furthermore, for the shift operation, we could use shift registers. Still, instead, we preferred to implement using the normal registers as using the shift registers would have created complications regarding the operations of other main functions. Or else we could have used more registers for shifting but we are required to operate on only the given 4 registers. Thus, keeping it to the normal registers gave us the choice of implementing multiple functions using the same registers.

Preliminary Design



Design Description

Display Block:

- The MCLU sends the final result to this block.
- On a screen or other output device, the outcome is displayed.
- The flowchart is over when the outcome is shown.

Arithmetic Block (Complete Addition and Subtraction):

- This block accepts values as inputs from Registers 1 and 2.
- Depending on the desired operation, it either adds or subtracts from the input values.
- The arithmetic operation's outcome is then forwarded to the MCLU for additional processing.

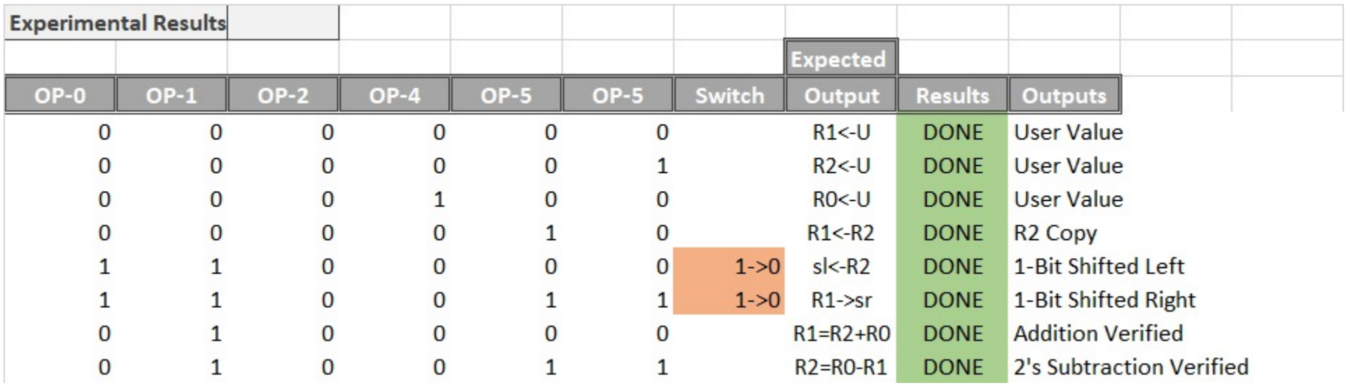
Execute a shifting block:

- The value from Register 3 is the input for this block.
- Depending on the desired shift direction, it performs a shifting operation on the input value to the left or the right.
- After that, the shifted value is returned to the MCLU for additional processing.

User Data Block:

- The user is prompted to enter a value in this block.
- It is patiently awaiting input from the user.
- The MCLU receives the entered value and processes it further.

Experimental Results



Experimental Results

							Expected		
OP-0	OP-1	OP-2	OP-4	OP-5	OP-5	Switch	Output	Results	Outputs
0	0	0	0	0	0		R1<-U	DONE	User Value
0	0	0	0	0	1		R2<-U	DONE	User Value
0	0	0	1	0	0		R0<-U	DONE	User Value
0	0	0	0	1	0		R1<-R2	DONE	R2 Copy
1	1	0	0	0	0	1->0	sl<-R2	DONE	1-Bit Shifted Left
1	1	0	0	1	1	1->0	R1->sr	DONE	1-Bit Shifted Right
0	1	0	0	0	0		R1=R2+R0	DONE	Addition Verified
0	1	0	0	1	1		R2=R0-R1	DONE	2's Subtraction Verified

Performance Analysis

Initially, we set the first 2 bits from MSB to 00. It resultantly enabled the decoder responsible for the Transfer operation. As we set the remaining 4 bits from MSB to 0000, it, as expected, performed the first sub-operation i.e. loading the inputs, we manually provided in 4-bit binary, in the R1 register. There was no error in the process and the probes showed correct readings. Similarly, when we changed the 4 bits to 0001, the inputs we provided were loaded in the R2 register. No faults were apparent. Likewise, we performed the third sub-operation and last sub-operation. No apparent fallacies were found while testing our MCU. Even in the fourth sub-operation, the data successfully moved from R2 to R1.

Secondarily, we set the first 2 bits from MSB to 01. That enabled the decoder responsible for Arithmetic operations. As we set the remaining 4 bits to 0000, it performed the add sub-operation and added the 4-bit binary data in R2 and R0 and stored it in R1. The probes showing the output of R1 clearly conveyed that the addition operation was performed successfully. Moreover, we also set the 4 bits from the OP code to 0011, which performed the second sub-operation i.e. subtraction. This operation performed a binary subtraction using 2's complement. It subtracted the data in R1 from R0 and stored it in R2. The desired outputs from R2 were visible.

At last, we set the first 2 bits from MSB to 11. It enabled the decoder responsible for the Shift operation. As we set the remaining 4 bits to 0000, the readings on the probes, showing R2 outputs, started to shift with the pattern of N-1. This showed that a Right shift sub-operation was achieved as intended. In addition, after setting the 4 bits to 0011, the reading on the probes connected to the output of R1 showed a Left shift i.e. in the pattern N-1.

No errors were apparent from the testing as we operated all the functions multiple times with different values. Only the desired outputs were shown by the probes.

Conclusion

In fine, we designed a Mini-Computational Unit which essentially performed 3 major tasks involving registers R1, R2, R3, and a flag register. In the following design, we employed the idea of using a 6-bit binary OP code that enabled the data inputs to control the data and similarly used the decoders to control its flow. The first main function is to load the binary data input in the specified registers. The second main function performs the Shift operations. With the OP code provided, it either shifts left or right. The last main function executes arithmetic sub-operations. With the specified OP code, it either performs addition or 2’s complement subtraction. There is no overflow of data nor any errors in the operations.