Zeynepnur ŞAHİNEL
2305399 08.05.2022

EE446 EXPERIMENT 3 PRELIMINARY WORK
SINGLE-CYCLE PROCESSOR DESIGN

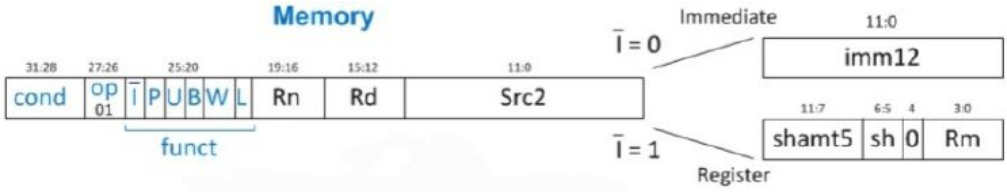
Datapath Design
Part 1

For the instructions in the CPU that you are going to design, list all the steps that are needed for the execution of each instruction. While adding each instruction show all of the changes in the connections of the datapath.

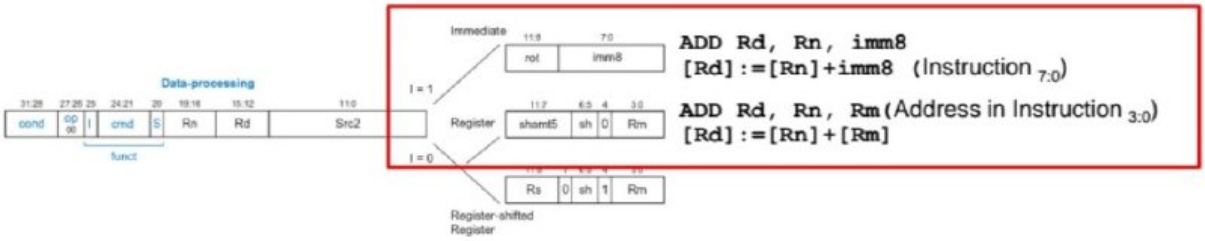
-Instruction Fetch:

Before the clk edge comes, the program counter (PC) is ready at the clk register input. At the rising edge clk, PC is assigned as the output of the register and the input of the Instruction Memory. All read operations are combinational, no clk is needed. Thus, at the output of the instruction memory, related instruction appears. (32-bit)

Memory Instruction



Data Process Instruction



-Read source operands from the register file:

With Register File, the write and read operation is handled. There are 2 reads and 1 write port. Writing operation works with clk, whereas read is operational. Here, A1 is loaded with Rn (19:16 bits), and A2 is connected to 2to1 MUX of Rd (15:12) and Rm (3:0). Moreover, when the write operation occurs, the data coming from the WD3 port is written to A3.

-Zero-extend the immediate:

Immediate extender extends the instruction (11:0) with zeros. (31:12)

-Compute the memory address:

Data memory handles memory read and memory write. Input A is responsible for the address input.

-Read data from the memory and write it back to the register file:

In the data-processing and LDR operation register is written with data inside the data memory.

Zeynepnur ŞAHİNEL
2305399 08.05.2022

-Determine the address of the next instruction:

PC is determined with either PC+4 or target branch address. These inputs are seperated with a 2 to 1 MUX.

1-ADD

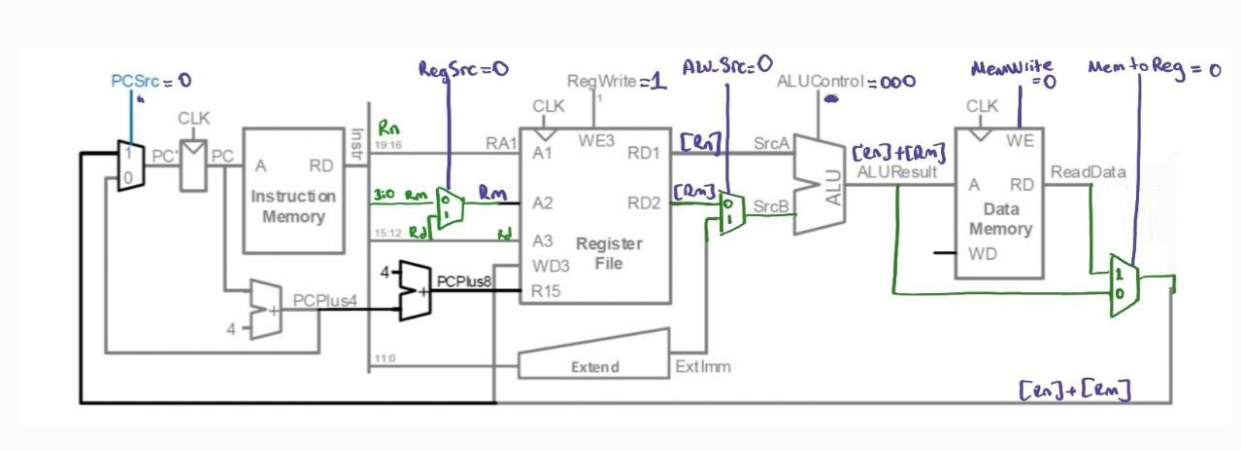


Figure 1. Datapath of ADD operation

2-SUB

Datapath is the same as in Figure 1. Only ALUControl Signal=001

3-AND

Datapath is the same as in Figure 1. Only ALUControl Signal=100

4-ORR

Datapath is the same as in Figure 1. Only ALUControl Signal=101

5-LSR

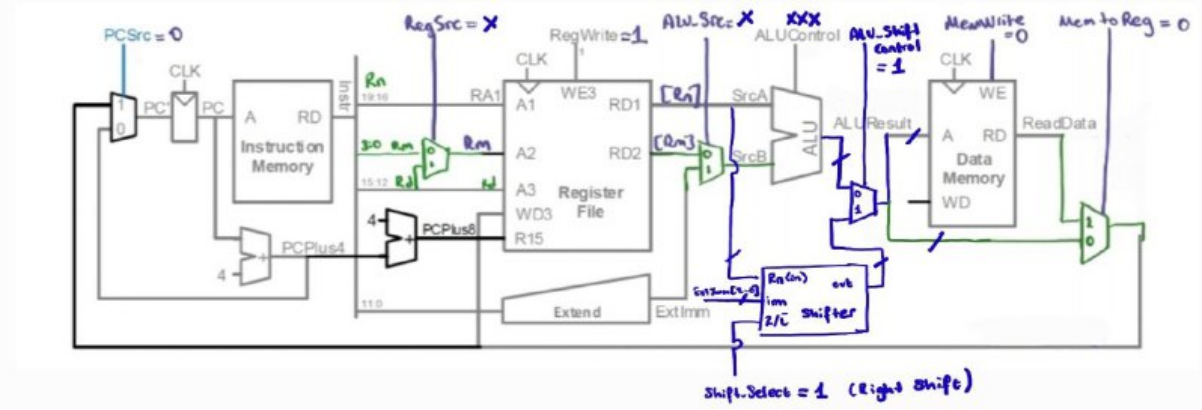


Figure 2. Datapath of the LSR Operation

6-LSL

Datapath is the same as in Figure 2. Only Shift_Select=0

7-CMP

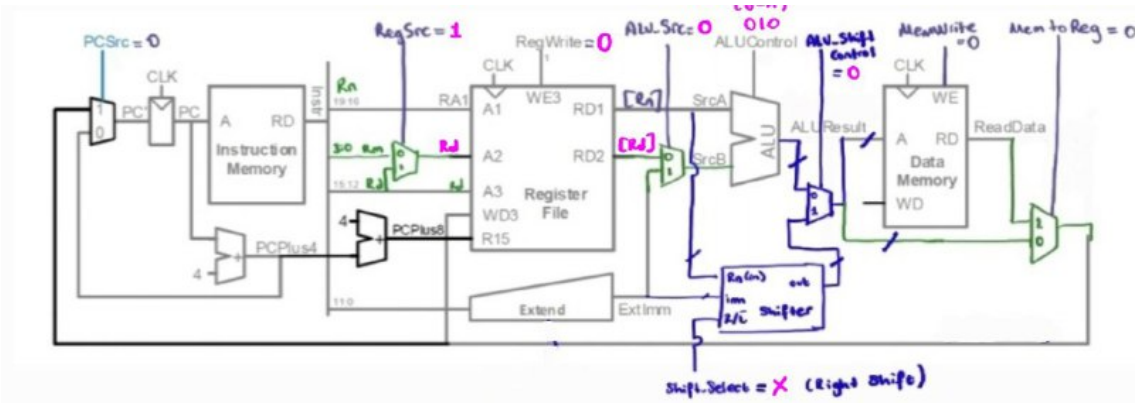


Figure 3. Datapath of the CMP operation

Here, the subtraction result is not written to the destination register. Only, according to the result of the Z flag (one of the ALU Flags), the CPSR is updated.

8-STR

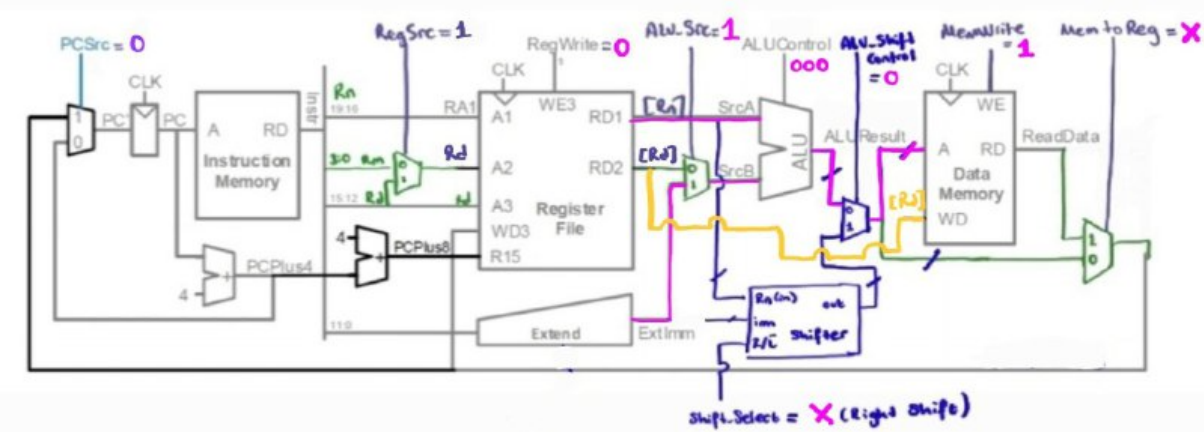


Figure 4. Datapath of the STR operation

9-LDR

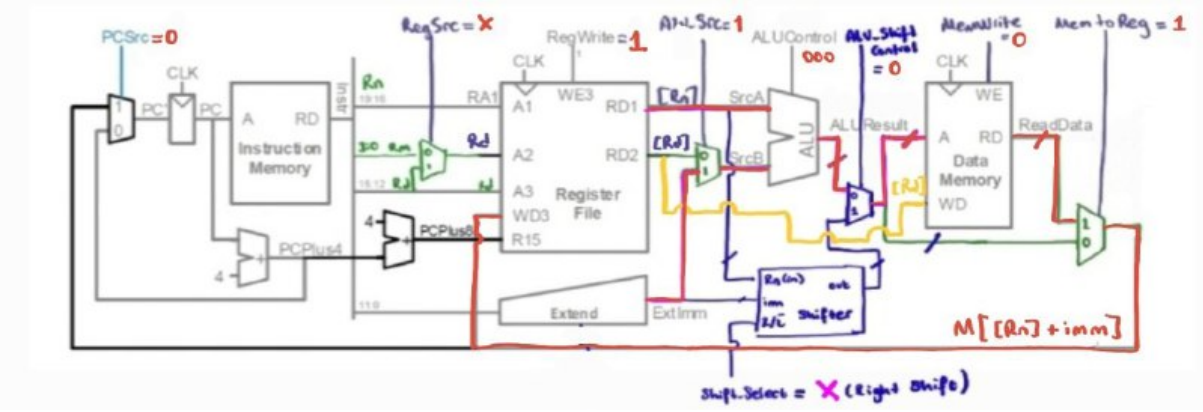


Figure 5. Datapath of the LDR operation

Part 2

State the control signal inputs of your overall design. Draw a black box diagram of your architecture by indicating the inputs and outputs

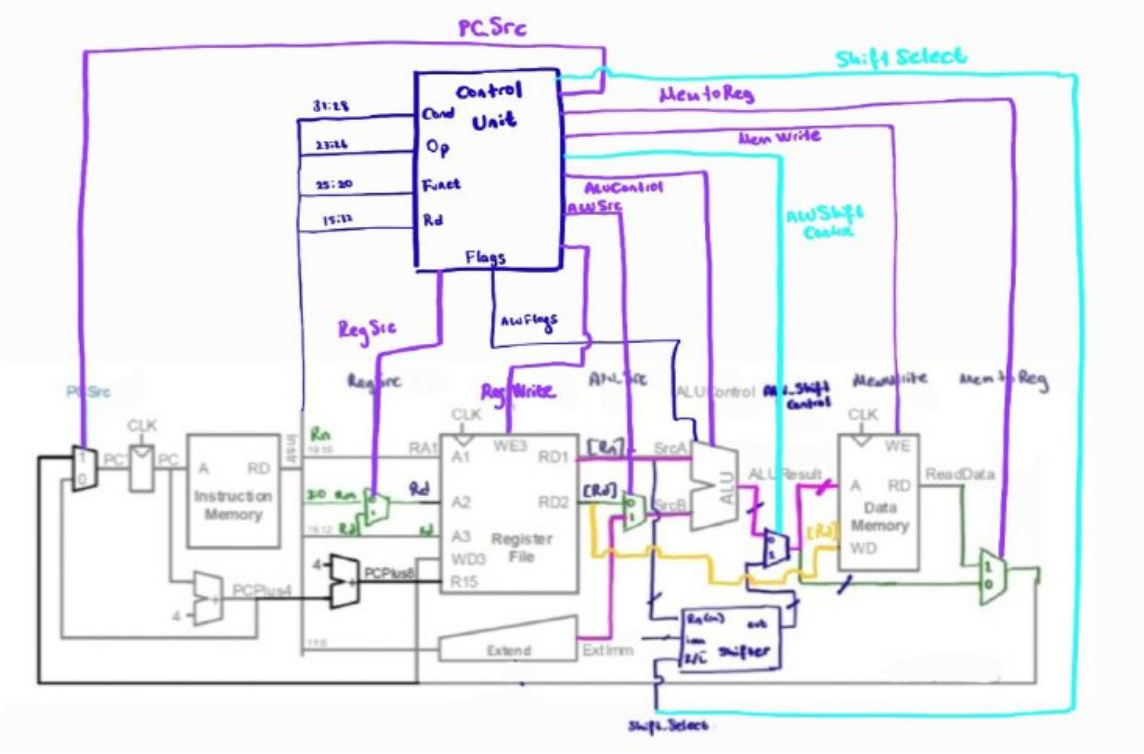


Figure 6. Overall Design

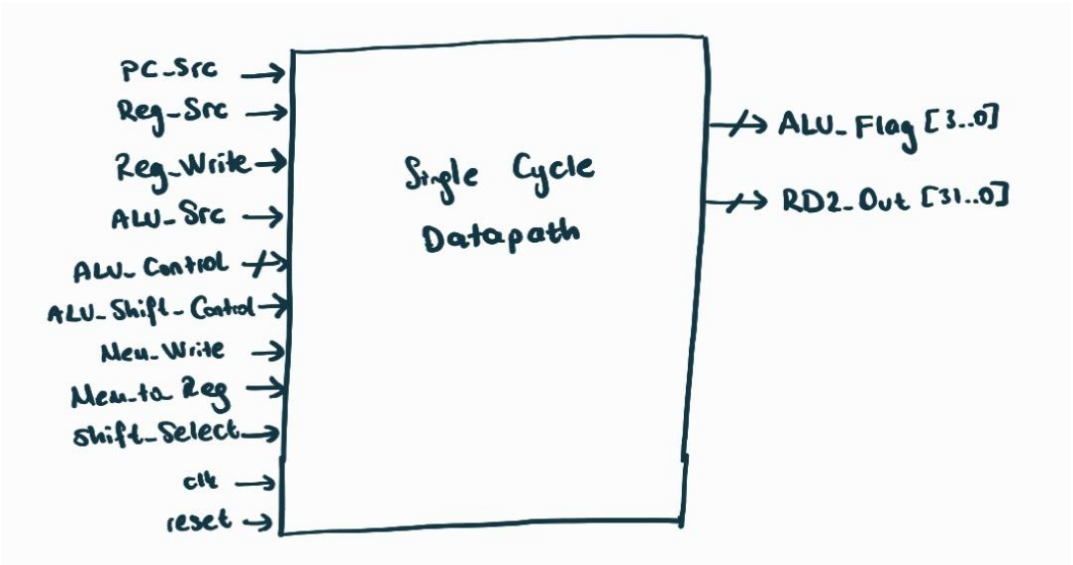


Figure 7. Black Box Diagram of the Datapath

Part 3

Implement your datapath design in Schematic Editor of Quartus

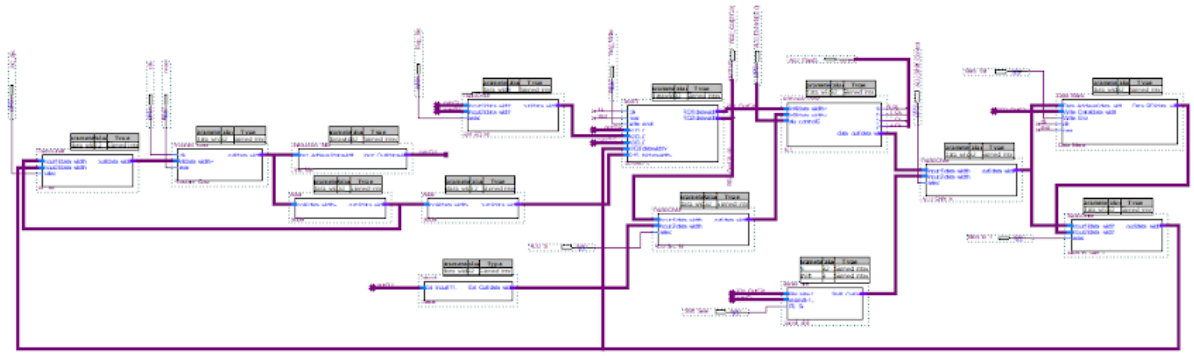


Figure 8. Datapath Schematic

Controller Design

Part 1

Draw the controller unit as a black box diagram and indicate its inputs and outputs.

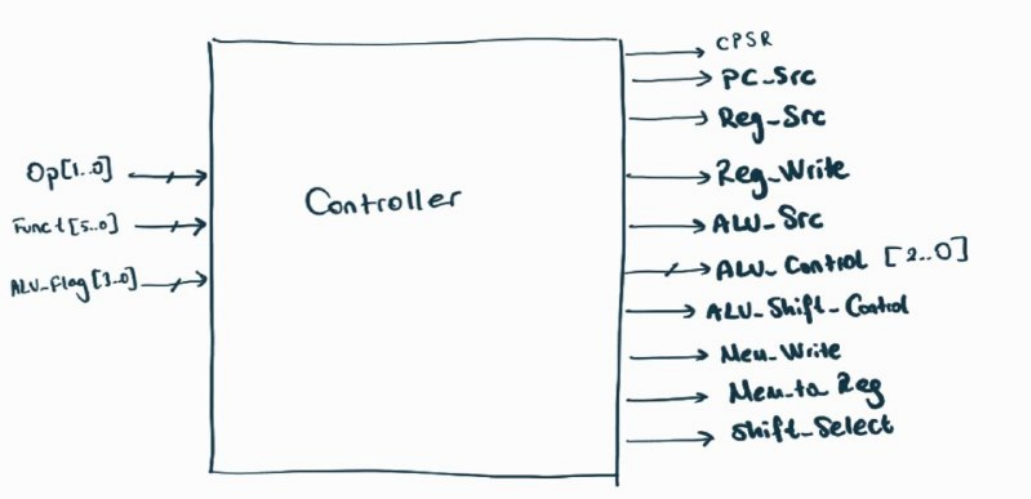


Figure 9. Black Box Diagram of the Controller

Part 2

While adding each instruction show all of the changes in the control signals.

Table 1. Operations and Their Control Signals Step by Step

	ADD	SUB	AND	ORR	LSR	LSL	CMP	STR	LDR
PC_Src	0	0	0	0	0	0	0	0	0
Reg_Src	0	0	0	0	X	X	1	1	X
Reg_Write	1	1	1	1	1	1	0	0	1
ALU_Src	0	0	0	0	X	X	0	1	1

ALU_Control	000	001	100	101	XXX	XXX	010	000	000
ALU_Shift_Control	-	-	-	-	1	1	0	0	0
Mem_Write	0	0	0	0	0	0	0	1	0
Mem_to_Reg	0	0	0	0	0	0	0	X	1
Shift_Select	-	-	-	-	1	0	X	X	X

Part 3

Give the truth table for the main controller of the single-cycle processor.

Table 2. Truth Table for the Controller

Op	**Funct**	**Type**	**PC_Src**	**Reg_Src**	**Reg_ Write**	**ALU _Src**	**ALU_ Control**	**ALU_ Shift_ Control**	**Mem_ Write**	**Mem_to _Reg**	**Shift_ Select**
00	101000	ADD	0	0	1	0	000	0	0	0	x
00	100100	SUB	0	0	1	0	001	0	0	0	x
00	100000	AND	0	0	1	0	100	0	0	0	x
00	111000	ORR	0	0	1	0	101	0	0	0	x
00	100110	LSR	0	x	1	x	xxx	1	0	0	1
00	100010	LSL	0	x	1	x	xxx	1	0	0	0
00	100101	CMP	0	1	0	0	010	0	0	0	x
01	011000	STR	0	1	0	1	000	0	1	x	x
01	011001	LDR	0	x	1	1	000	0	0	1	x

Part 4

Implement your controller in Verilog HDL.

Code was uploaded to ODTUCLASS.

Usage of Parameters

Why is it important to use parameterized design?

Using parameters provides reusability of same modules for different cases.

Is it plausible to use parameters for the data-width in this laboratory?

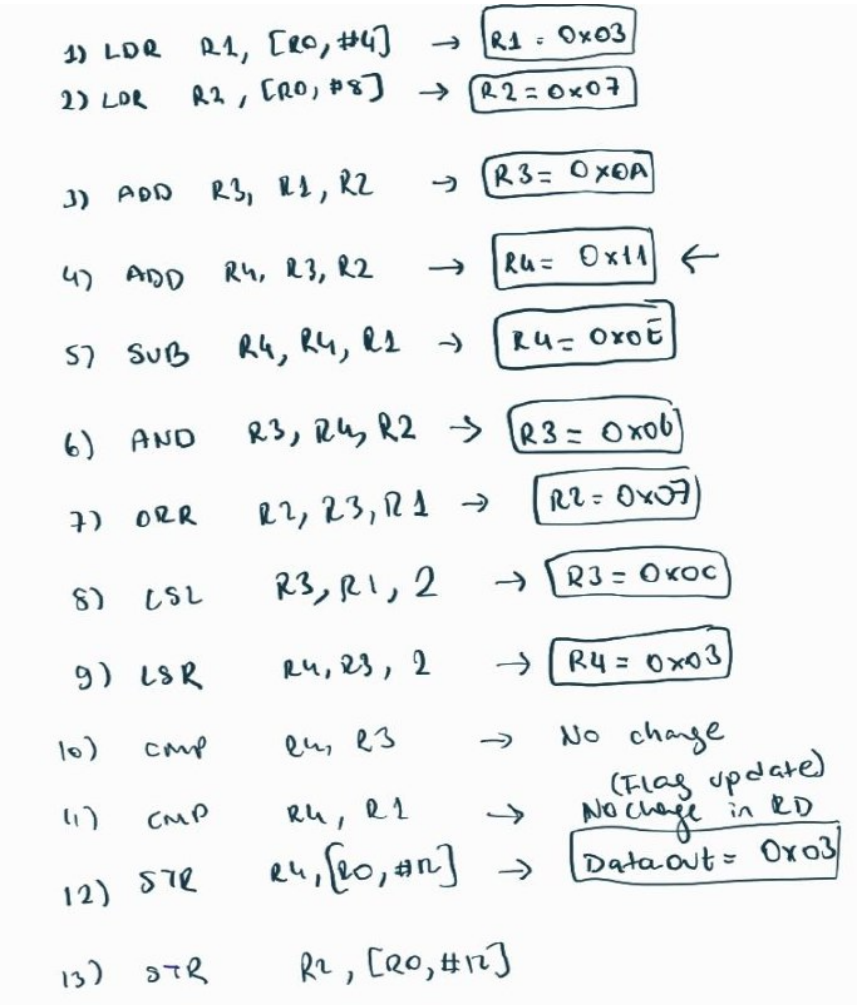
Yes, it is plausible since in some cases, we have different data-width for the same module. For instance, in the RM_RD_MUX inputs are 4 bits whereas other 2 to 1 MUX inputs are 32 bits.

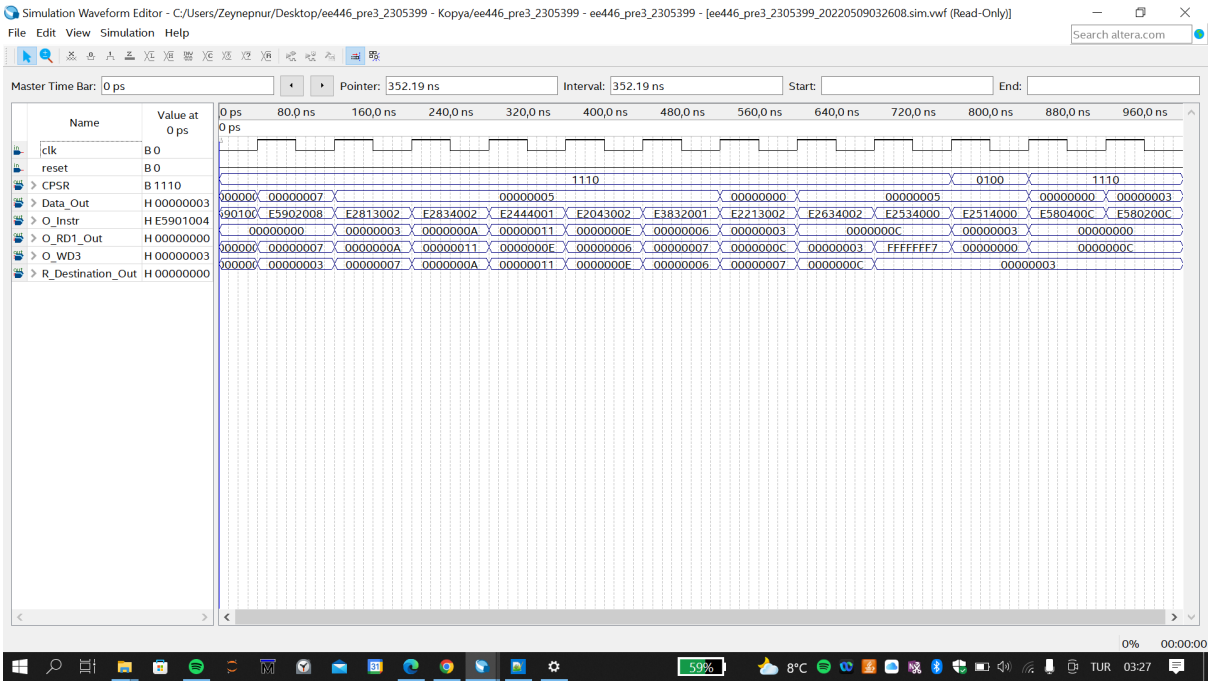
What are the possible complications?

Would the design work with an arbitrary data-width?

It would work, but all data-width parameters should be compatible.

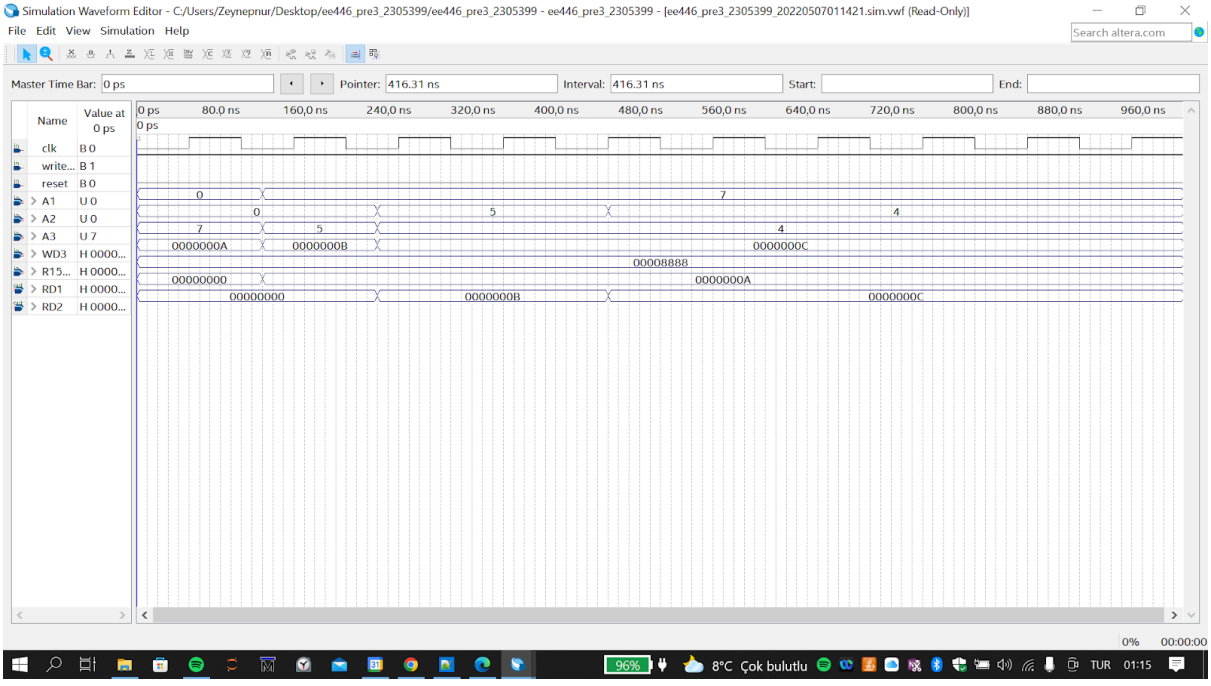
Experimental Work



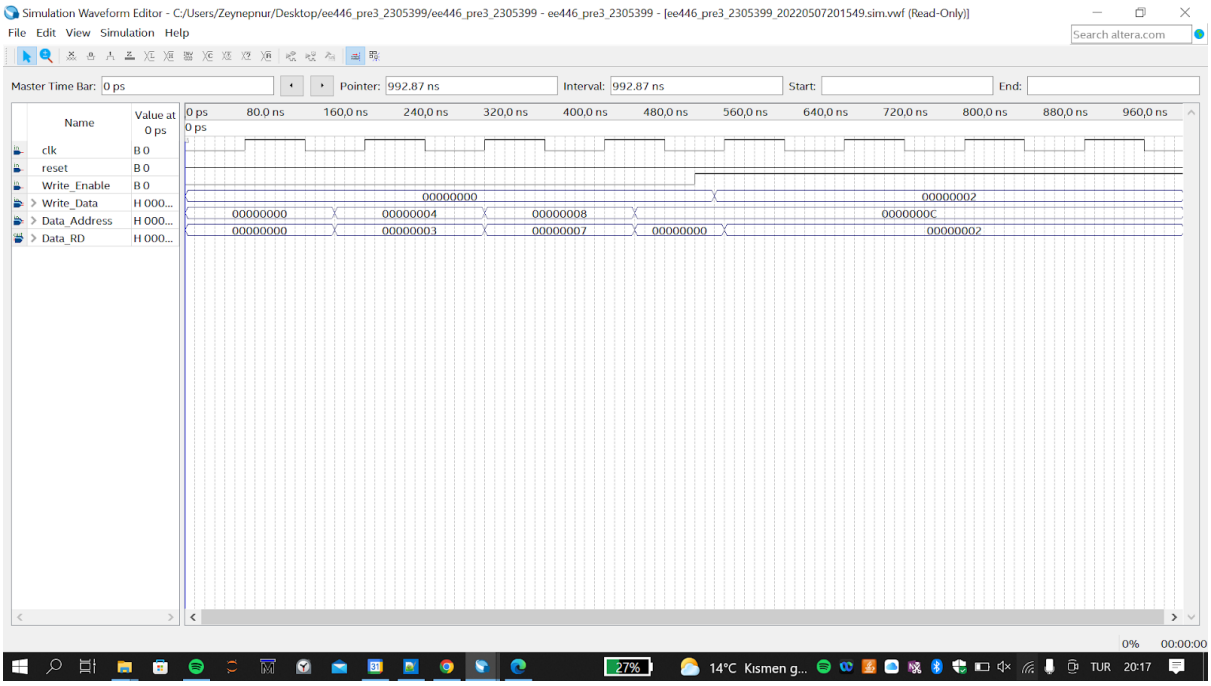


Top Module Waveform

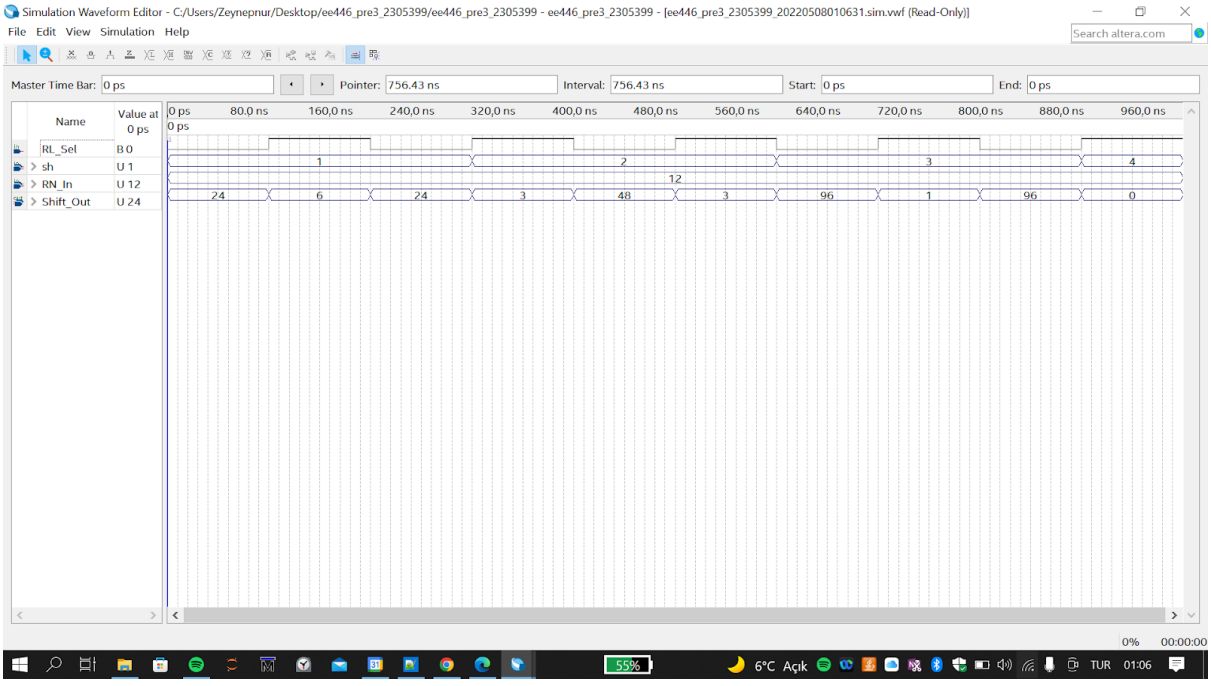
Appendix



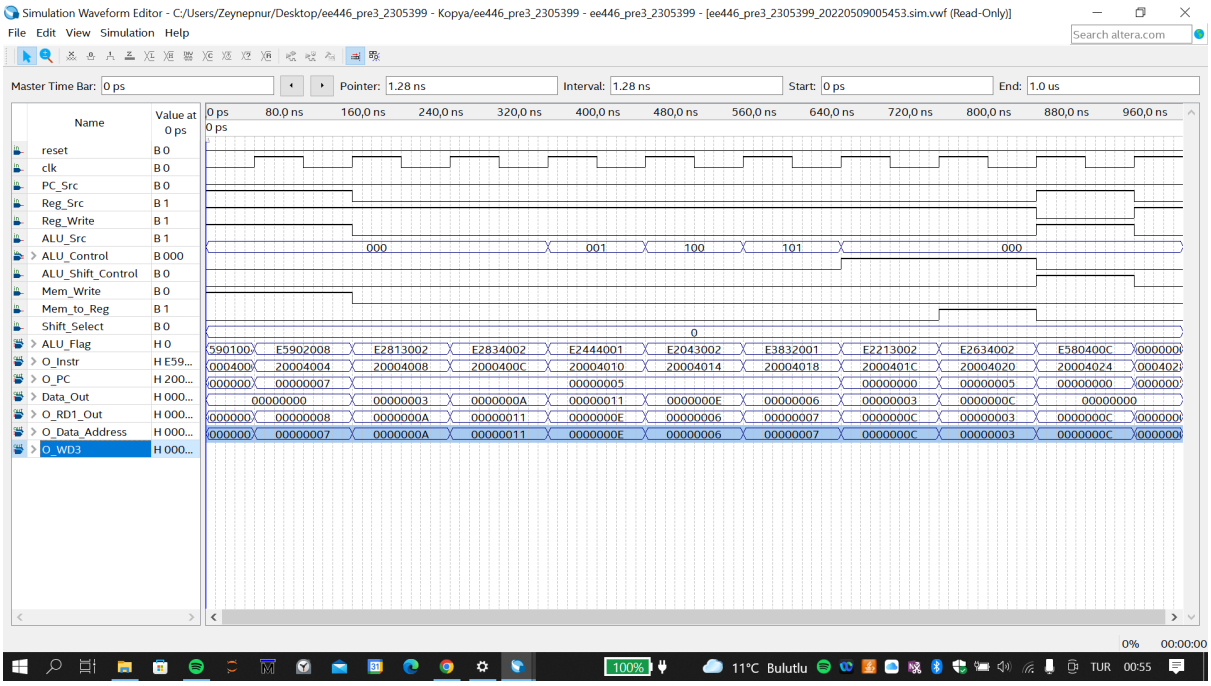
Register File Waveform Example



Data Memory Waveform Example



Barrel Shifter Waveform Example



Datapath Waveform without combined with Controller