METU EE 446
Computer Architecture Laboratory

Arithmetic Logic Processor Design

I'm looking for something really dull and repetitive.
Can you control ALU?
Resume
Name: ASM
Experience: In-class exercises
JOBS! JOBS! JOBS!

Laboratory Work 2 - Arithmetic Logic Processor Design

Objectives

The purpose of this laboratory work is to practice the design of a simple processor. In that manner, you will design and implement datapath and control unit of an arithmetic logic processor. The control unit is to be designed as an algorithmic state machine (ASM). The designed processor architecture will be able to take two inputs and perform multiplication, division and combinatorial arithmetic logic unit (ALU) operations on the inputs.

During this laboratory work, you will improve your hard-wired controller design skills by designing the controller unit. In this way, you will be familiar with the controller design of processor architectures. Additionally, you will also practice datapath design while constructing your own architecture and improve your design skills that will help you in future laboratory works. Finally, you will embed your design to the FPGA of DE0-Nano board and experiment it.

1 Preliminary Work

To fulfill the requirements of this laboratory work, the following tasks should be performed.

1.1 Reading Assignment

The laboratory manual where the regulations and some other useful information exist is available at the ODTUClass course page. Read that manual thoroughly. If you feel yourself unfamiliar with **ASM** design and **datapath** design, please refer to the corresponding lecture notes of EE445 course.

1.2 Arithmetic Logic Processor (100% Credits)

For this laboratory work, you will design and implement an 8-bit arithmetic logic processor (ALP). First, you will design its datapath then you will implement corresponding controller.

The ALP you will design has 5 input ports and 3 output ports. There is a data input port for 8-bit input data and a 3-bit input port for operation select (OP). Additionally, there are three control inputs for data load (LOAD), content clearing (CLR) and computing (COMP). At the output, there are two 8-bit outputs for result and an error output pin (ERR) to indicate the ALP operation has resulted in an arithmetic overflow. The ALP as a black box is depicted in Figure 1.

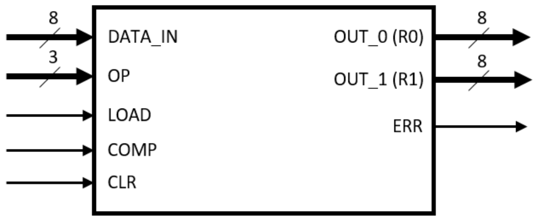

Figure 1: Black box diagram of the arithmetic logic processor (ALP) module

Now, the specifications of the ALP is to be explained. There are two registers R0 and R1. The input data is to be loaded to those registers and the result will be stored also in those registers. Together with the control signals, the operation of the ALP should be as follows:

- With the **LOAD** signal, the data at the input port is loaded to R0 and the content of R0 is loaded to R1. **LOAD** signal has no effect during computation.
- With the **COMP** signal, the result of the operation which is determined by **OP** signal is computed for the contents of R0 and R1.
- With **CLR** signal, the contents of the registers are cleared. **CLR** signal has no effect during computation.
- For the addition, subtraction and logic operations; the result is stored in R0 and the content of R1 is cleared.
- For the multiplication operation, the most significant byte of the result is stored in R1 and the least significant byte is stored in R0.
- For the division operation, the quotient is stored in R0 and the remainder is stored in R1.
- For the addition, subtraction and division operations, **ERR** output is 1 if the corresponding operation results in an arithmetic overflow.
- For the arithmetic operations, the numbers are signed and they are represented in 2's complement form.

The specifications of the ALP is summarized in Table 1 and the operations that the ALP support is provided in Table 2.

Table 1: ALP Description

Control Signal / Output	**Description**

LOAD	if 1, $R0 \leftarrow$ DATA_IN, $R1 \leftarrow R0$;
	if 0, the registers R0 and R1 retain their content;
	has no effect during computation of an operation

COMP	if 1, the computation of the operation specified by OP begins with the current content of the R0 and R1 registers and the result is written to the corresponding register; has no effect during computation of an operation
CLR	if 1 the contents of the R0 and R1 registers are cleared; has no effect during computation of an operation
ERR	1 if the add, subtract or division operation result in overflow

Table 2: ALP Operation Control

OP [2:0]	**ALP Operation**	**Explanation**
000	Addition	$R0 \leftarrow R0 + R1$, $R1 \leftarrow 0$
001	Subtraction	$R0 \leftarrow R0 - R1$, $R1 \leftarrow 0$
010	Multiplication	$\{R1, R0\} \leftarrow R1 \times R0$ (signed 2's complement)
011	Division	$R1/R0$, $R0 \leftarrow$ Quotient (signed 2's complement), $R1 \leftarrow$ Remainder
100	AND	$R0 \leftarrow R1 \wedge R0$, $R1 \leftarrow 0$
101	OR	$R0 \leftarrow R1 \vee R0$, $R1 \leftarrow 0$
110	EXOR	$R0 \leftarrow R1 \oplus R0$, $R1 \leftarrow 0$
111	Bit Clear	$R0 \leftarrow R1 \wedge \neg R0$, $R1 \leftarrow 0$

Note that for the following steps of the preliminary work, there is **no unique** solution. Therefore, you can come up with your own unique design. You are strongly encouraged to work on your own and try to design everything by yourself. This is the key to improve yourself in computer architecture design and become eligible engineers when you will graduate. Moreover, the **efficiency** of your design will bring you **bonus** credits. Thus, try to minimize the complexity of your designs.

For each item in the following steps, you should submit hard copy of your work including explanations.

1.2.1 Datapath Design (30% Credits)

You are given an initial architecture in Figure 2. There are two registers with write enable control. Those are for R0 and R1 register. Additionally, there are two shift registers and a 2-bit status register. Finally, there is the combinatorial ALU which you have designed in the scope of the first laboratory work. You will complete the datapath step by step so that the specifications given in Section 1.2 are met. You can use your modules in your library which has been constructed in the scope of the first laboratory work. Additionally, you can add additional registers of any size (1-bit, 2-bit, ...) if you need.

Considering the architecture provided in Figure 2, perform the following design steps:

1. (5% Credits) Design a datapath so that the 2's complement of the number stored in the R0 or R1 register is taken and stored back to the corresponding register with a proper control signal. In other words, if R0 has X as the content, then with a proper control signal, the content of the R0 should be updated to $-X$. Similarly, if R1 has Y as the content, then with a proper control signal, the content of the R1 should be updated to $-Y$, in 2's complement form.
2. (2% Credits) Extend your datapath so that the 2's complement of the number in a register is to be taken if the number is negative. Indicate whether you should provide any input signals to the controller.

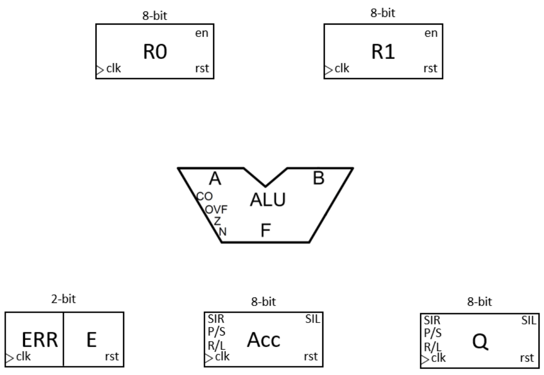

Figure 2: Architecture to which a datapath is to be designed

3. (3% Credits) Extend your datapath to support the ALP operations except for multiplication and division.
4. (6% Credits) Explain your signed multiplication algorithm and extend your datapath to support signed multiplication operation. Indicate whether you should provide any input signals to the controller and explain.
5. (7% Credits) Explain your signed division algorithm and extend your datapath to support signed division operation. Indicate whether you should provide any input signals to the controller.
6. (2% Credits) Finally, state the control signal inputs of your overall design and the outputs from your design to the controller as the inputs. Draw a black box diagram of your architecture by indicating the inputs and outputs.
7. (5% Credits) Implement your datapath design in Schematic Editor of Quartus.

1.2.2 Controller Design (70% Credits)

In this step, the controller for the ALP is to be designed as ASM. For the sake of clarity, we will consider 4 black boxes each of which is a part of the ASM chart of the controller. First box is the part of the ASM chart until **COMP** signal is asserted. The second box is for the ALP operations except for multiplication and division. The third and the forth boxes are for multiplication and divison, respectively. You will step-by-step design those boxes.

1. (5% Credits) Draw the controller unit as a block box diagram and indicate its inputs and outputs.
2. (3% Credits) Draw the ASM chart of the first box and leave the rest of the controller as black boxes.
3. (2% Credits) Draw the ASM chart of the second box and connect it to the first box.
4. (10% Credits) Draw the ASM chart of your signed multiplication algorithm and connect it to the first box. Explain how you handle signed numbers.
5. (15% Credits) Draw the ASM chart of your signed divison algorithm and connect it to the first box. Explain how you handle signed numbers.
6. (2% Credits) Considering the overall design, how many cycles does it take to compute the result of each operation after the **COMP** signal is asserted.
7. (3% Credits) How many states does your ASM have? Assuming that the **OP** signal remains unchanged during the computation, discuss whether you can benefit from that constraint.
8. (30 % Credits) Implement your controller in Verilog HDL.

2 Experimental Work

2.1 Arithmetic Logic Processor (100% Credits)

Load your 8-bit ALP designed in the Preliminary Work Part 1.2 to the DE0-Nano board. Use 8 switches in the DEES as the data input, use 3 switches of the DE0-Nano board as the operation select. For load, compute and clear signals, you can use push buttons from either board. For the outputs, use 8 LEDs of DEES for the R0 register and use 8 LEDs of the DE0-Nano board for the R1 register. For error indicator, you may use any pin of the 8-segment displays of DEES.

1. Prepare examples to be tested to verify the correct operation of your design.
2. Verify the operation of your design by performing the example operations and demonstrate it to your lab instructor.

Note that partially working designs are to be penalized. The performance grading for the partial work is:

1. Only input part: 5% Credits
2. Can take and store the 2's complement: 10% Credits
3. Multiplication and Division excluded: 30% Credits (No error signal: -5% Credits)
4. Division excluded: 60% Credits (Can not handle signed numbers: -15% Credits)
5. All operations but no signed operations: 70% Credits

!!! Important Notice !!! When using DEES in your experiments, **DO NOT CONNECT VCC pins of DE0-Nano Board** to DEES's VCC terminal. Also, to make a common voltage reference, CONNECT GND of DE0-Nano Board to DEES's GND terminal. Please make sure that you turn all supplies OFF while making any connection throughout laboratory work.

3 Parts List

```
DE0-Nano Board
DEES
Oscilloscope
```