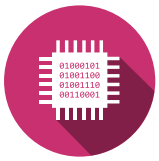


Arithmetic and Logical Unit

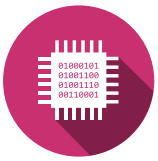
Labor Digital Design

Contents

1 Goal

This lab aims to practice the design of logical circuits using multiplexers. It provides a method for implementing arithmetic and logical units (Logical Unit (LU) and Arithmetic Unit (AU)) for microprocessors.

In a first lab session, the LU and AU are realized. In a second session, a complete Arithmetic and Logical Unit (ALU) is built, combining the two units and selecting the desired result using appropriate control signals.



2 Logical Unit LU

The Figure 1 shows the circuit of a Logical Unit (LU) of a microprocessor. The logical operations are performed bit by bit. Eight of these blocks form an 8-bit LU .

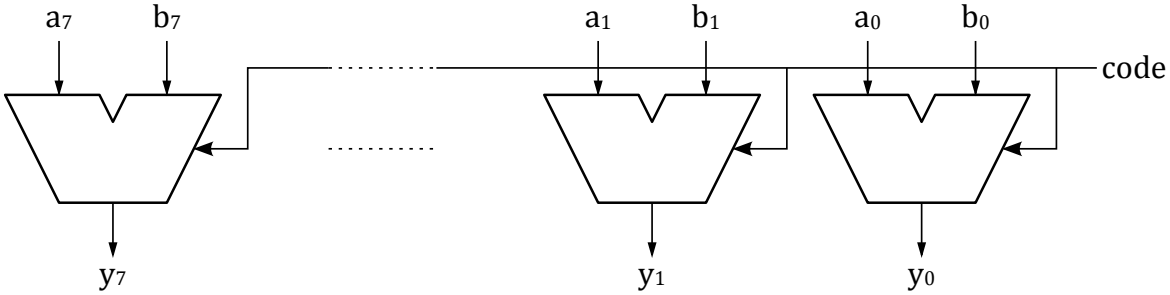


Figure 1 - Logical Unit LU

The logical circuit of each iterative block is realized with multiplexers, which represent a truth table. The control inputs $\text{sel}_0 = a_i$, $\text{sel}_1 = b_i$ and $(\text{sel}_3, \text{sel}_2) = \text{code}[1:0]$ are used to determine the function to be generated.

Establish the truth table of the logical function that produces the following operations in the programmable logic block:

- $y_i = b_i$ for **`code = "00"`** Load value b
- $y_i = a_i * b_i$ for **`code = "01"`** AND function between a and b
- $y_i = a_i + b_i$ for **`code = "10"`** OR function between a and b
- $y_i = a_i \oplus b_i$ for **`code = "11"`** XOR function between a and b

2.1 Implementation and Simulation

Implement the circuit of the LU of Figure 1. Some circuit elements are already given in the block **`ALU/LU1`**. Complete the missing input connection from the Multiplexer which should be connected either be a logical **`"0"`** or logical **`"1"`**. These values can be generated by the elements **`gates/logic0`** respectively **`gates/logic1`**.

The Testbench **`ALU_test/LU8_tb`** is already provided but does not test all cases. The testbench has to verify the functionality of the **entire** LU.

Complete the circuit of the iterative block of the **`ALU/LU1`** which performs the 4 specified operations.

Complete the test stimuli **`ALU_test/LU8_tester`** and verify the function of the entire **`ALU_test/LU8_tb`** with the simulation file **`$SIMULATION_DIR/ALU1.do`**.

3 Arithmetic Unit AU

The Figure 2 shows the iterative circuit of an Arithmetic Unit (AU). Also this circuit consists of eight 1-bit units connected to create a 8-bit AU. The logic circuit will be realised with multiplexers, which represent a truth table.

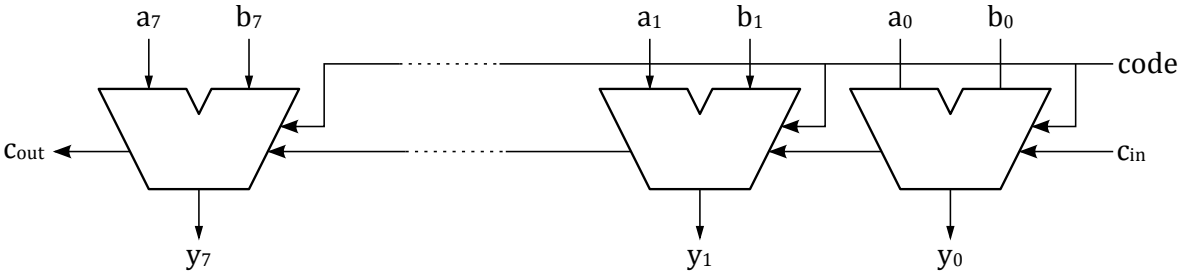


Figure 2 - Iterative Arithmetic Unit

The function will be realized with a multiplexer whose control signals are $\text{sel}_0 = a_i$, $\text{sel}_1 = b_i$, $\text{sel}_2 = c_{\text{in}}$ and $\text{sel}_3 = \text{code}[0]$.

> Write the truth table of the iterative logic block **`ALU/AU1`** of Figure 2 for each of the following operations on integers:
> - $y = a + b + c_{\text{in}}$ for **`code[0] = '0'`** ⇨ addition
> - $y = a - b - c_{\text{in}}$ for **`code[0] = '1'`** ⇨ subtraction

Considering that a left shift corresponds to a multiplication by two, propose an extension to the iterative circuit to realize the left shift function:

$$y = a \ll 1 = 2a = a + a \text{ for } \texttt{code[1:0] = '10'}$$

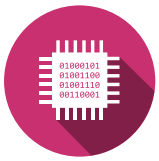
> Extend the circuit **`ALU/AU1`** to support the shift left operation when the value of **`code[1:0] = '10'`**.

3.1 Implementation and Simulation

Implement the circuit of the AU of Figure 2 to cover all functionalities mentioned before. Some circuit elements are already given in the block **`ALU/AU8`**.

The Testbench **`ALU_test/AU8_tb`** is already provided but does not test all cases. The testbench has to verify the functionality of the **entire** AU.

> Complete the circuit of the iterative block of the **`ALU/AU1`** which performs the 3 specified operations.
>
> Complete the test stimuli **`ALU_test/AU8_tester`** and verify the function of the entire **`ALU_test/AU8_tb`** with the simulation file **`$SIMULATION_DIR/ALU2.do`**.



4 Checkout Part 1

This is end of the first part of the labo, you have successfully built an Logical Unit and Arithmetic Unit. Before leaving the laboratory, ensure you have completed the following tasks:

- [ ] Circuit Design
  - [ ] Verify that the blocks **`ALU/LU8`** and **`ALU/AU8`** has been designed and tested with features mentioned in Section 2 and Section 3.
- [ ] Simulations
  - [ ] The specific tests of the respective testbenches (**`ALU_test/AU8_tb`** and **`ALU_test/LU8_tb`**) have been adapted to the circuit and ensures a complete test.
  - [ ] The circuits have been successfully tested with the respective testbenches **`ALU_test/AU8_tb`** and **`ALU_test/AU8_tb`**.
- [ ] Documentation and Projectfiles
  - [ ] Ensure all steps (design, convertions, simulations) are well-documented in your lab report.
  - [ ] Save the project to a USB stick or the shared network drive (**`\\filer01.hevs.ch`**).
  - [ ] Share files with your lab partner to ensure work continuity.

5 Arithmetic and Logic Unit ALU

The Arithmetic and Logical Unit (ALU) is realized by combining the Logical Unit and Arithmetic Unit developed so far in Section 2 and Section 3, it also includes an additional *right shift* operation see Figure 3. It will support the assembler code of the Xilinx PicoBlaze instruction set.

In order for the ALU to perform the different operations, the control signals must be set accordingly. These are $\mathrm{LU}_{\mathrm{Code}}[1:0]$, $\mathrm{AU}_{\mathrm{Code}}[1:0]$, $\mathrm{select}_{\mathrm{AU}}$, $\mathrm{select}_{\mathrm{SR}}$, $c_{\mathrm{in\_AU}}$. These signals are listed in Figure 3 as well as Table 2.

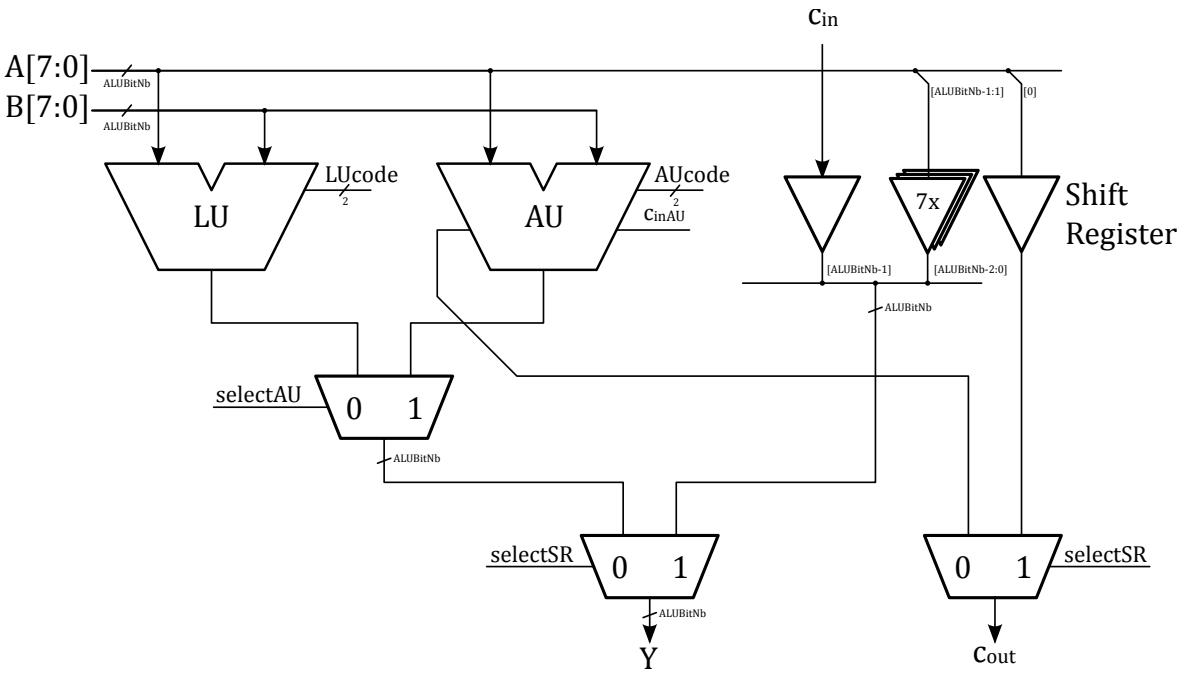
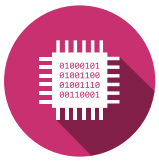


Figure 3 - Internal Structure ALU

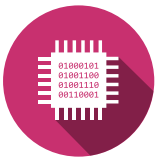
Investigate how the Shift Register in Figure 3 operates and for what Instructions it is used.



The different operations are listed in the Table 1. It defines which OpCode executes which operation. The OpCode is a 5-bit code located in the bits $I_{17} : I_{13}$ of the instruction word.

OpCode $I_{17} : I_{13}$	Assembler Instruction	ALU code Operation	Operation
00000	LOAD	LOAD B	$y = b$
00001	*ununsed*	-	-
00010	INPUT	LOAD B	$y = b$
00011	FETCH	LOAD B	$y = b$
00100	*ununsed*	-	-
00101	AND	AND	$y = a$ AND b
00110	OR	OR	$y = a$ OR b
00111	XOR	XOR	$y = a$ XOR b
01000	*ununsed*	-	-
01001	TEST	AND	$y = a$ AND b
01010	COMPARE	SUB	$y = a - b$
01011	*ununsed*	-	-
01100	ADD	ADD	$y = a + b$
01101	ADDCY	ADDCY	$y = a + b + c_{\text{in}}$
01110	SUB	SUB	$y = a - b$
01111	SUBCY	SUBCY	$y = a - b - c_{\text{in}}$
10000	SH / ROT	SHR	$a \gg 1$
10001	SH / ROT	SHL	$a \ll 1$
10010	*ununsed*	-	-
10011	*ununsed*	-	-
10100	*non-ALU*	-	-
...	...	...	...
11111	*non -LU*	-	-

Table 1 - OpCode decoding in relation to the ALU operations



5.1 Implementation and Simulation

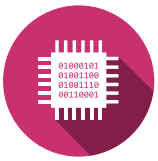
For each instructions all controlsignals in the circuit **ALU/ALU8** such as the multiplexers and the commands of the LU and AU are given in the Table 2. The truth table is based on the OpCode of Table 1.

code[4 : 0]	$LU_{code}[1:0]$	$AU_{code}[1:0]$	$select_{AU}$	$select_{SR}$	c_{in_AU}
00000					
00001					
00010					
00011					
00100					
00101					
00110					
00111					
01000					
01001					
01010					
01011					
01100					
01101					
01110					
01111					
10000					
10001					
10010					
10011					
10100					
...					
11111					

Table 2 - ALU Control Signals

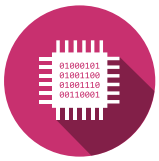
Complete the Table 2 which gives the values of all control signals based in the ALU Operation indicated with the signal code[4 : 0].

For each controlsignal deduce the boolean equation and implement it in the circuit **ALU/ALU8**.



Verify, and if necessary complete the test stimuli **`ALU_test/ALU8_tester`**. Run the testbench **`ALU_test/ALU8_tb`** with the simulation file **`$SIMULATION_DIR/ALU3.do`** and verify the ALU functionality.

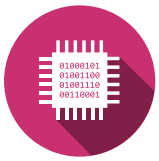
The individual operations such as **`ADD`**, **`OR`**, **`AND`**, ... were already verifed in the previous labo. And do not need to be fully verified again.



6 Checkout Part 2

This is end of the second part of the labo, you have successfully built an Arithmetic and Logical Unit of the Xilinx PicoBlaze. Before leaving the laboratory, ensure you have completed the following tasks:

- [ ] Understanding
  - [ ] You understand how the individual ALU components work, especially the new circuit of the Shift Register.
- [ ] Circuit Design
  - [ ] Verify that the block **`ALU/ALU8`** have been designed and tested with features mentioned in Table 1.
- [ ] Simulations
  - [ ] The specific tests of the testbench **`ALU_test/ALU8_tb`** has been adapted to the circuit and ensures a complete test.
- [ ] Documentation and Projectfiles
  - [ ] Ensure all steps (design, convertions, simulations) are well-documented in your lab report.
  - [ ] Save the project to a USB stick or the shared network drive (**`\\filer01.hevs.ch`**).
  - [ ] Share files with your lab partner to ensure work continuity.



Glossary

ALU **– Arithmetic and Logical Unit** 1, 5, 6, 7, 7, 8, 9

AU **– Arithmetic Unit** 1, 1, 3, 3, 3, 3, 7

LU **– Logical Unit** 1, 1, 2, 2, 2, 2, 7

PicoBlaze: PicoBlaze is a small, 8-bit microcontroller designed by Xilinx for use in FPGAs. It is often used in educational settings to teach basic microcontroller concepts. 5, 9