Digital Design and Computer Architecture: Lab Report		
Lab 5: Implementing an ALU		
Date	13.04.20	Grade
Names	Yann Huguet, Catja Käser	
		Lab session / lab room

You have to submit this report via Moodle.

Use a zip file or tarball that contains the report and any other required material. Only one member from each group should submit the report. All members of the group will get the same grade.

The name of the submitted file should be *Lab5_LastName1_LastName2.zip* (or *.tar*), where *LastName1* and *LastName2* are the last names of the members of the group.

Note 1: Please include all the required material. No links/shortcuts are accepted.

Note 2: The deadline for the report is a hard deadline and it will not be extended.

Exercise 1. Replacing the Adder with your Adder from Lab 2

In the lab Manual, you learned how to design an ALU and how to find out about its area. In this exercise, instead of using + for the adder in your ALU, use the adder you designed in Lab 2. To do so, you first need to build a 32-bit adder using the instances of the 4-bit adder you have built.

As you read in Chapter 5 of H&H book, hardware description languages provide the operation to specify a carry propagate adder. Modern synthesis tools select among many possible implementations, choosing the cheapest (smallest) design that meets the speed requirements. This greatly simplifies the designer's job.

Compare the area of the ALU in this exercise with the area you obtained from the ALU you designed in the manual and write about your conclusions based on this observation.

Note: For lab 6, you need to keep using the ALU you designed in the manual, not in this report.

Feedback

If you have any comments about the exercise please add them here: mistakes in the text, difficulty level of the exercise, or anything that will help us improve it for the next time.

The modification of the adder lowered both numbers associated with area of the ALU:

	Old		New	
LUT	135	0.65 %	38	0.18 %
IO	101	95 %	74	69.81 %

Conclusion: Synthesis tool has a hard time finding an efficient implementation. If we specify more precisely, how the implementation should be done, we get better performance.