Design of Digital Circuits: Lab Report

Lab 5: Implementing an ALU

Date	11.04.2019	Grade
Names	David Zollikofer	
	Sven Pfiffner	Lab session / lab room
		Fri:10-12 | E 26.3

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 *Lab1_LastName1_LastName2.zip* (or *.tar*), where *LastName1* and *LastName2* are the last names of the members of the group.

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.

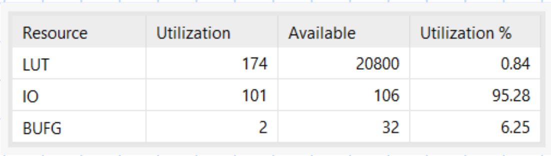
Observation

Implementing our own adder gives us the same functionality as using the addition of verilog.

However, one of the biggest advantages of HDL's such as Verilog is that they actually do a lot of work for us. In our particular example Vivado automatically choses the most efficient adder for our situation if we just use the + operator.

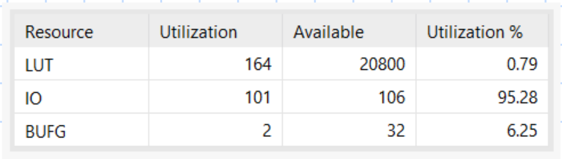
This can directly be seen when comparing the Elaborated Designs of the two implementations: Even though our design uses less LUT's than vivados (164 vs 174), vivados implementation uses a way faster adder than the ripple carry adder that we have used. (Refer to Harris & Harris p. 240 for details).

We think that it is generally a good idea to focus on the overall design choices and let Vivado do all the optimisation. Except of course, if one desires full control over the way a specific logic is implemented, for instance if we prefer small amounts of cells over efficency to create inexpensive circuits.



Resource	Utilization	Available	Utilization %
LUT	174	20800	0.84
IO	101	106	95.28
BUFG	2	32	6.25

Resource Utilisation of vivados adder



Resource	Utilization	Available	Utilization %
LUT	164	20800	0.79
IO	101	106	95.28
BUFG	2	32	6.25

Resource Utilisation of our own ripple-carry-adder.