Faculty of Science
Department of Computer Science
Chair for Embedded Systems
Prof. Dr. O. Bringmann

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Digital Design and Synthesis of Embedded Systems

Summer Term 2025

Exercise 8 : Deadline 13.07.2025 - 23:59 o'clock

The solutions are submitted via ILIAS as a packed archive (ZIP/TAR.GZ). This ZIP contains the written answers to questions, the source code (not as a listing in the PDF), the simulation results as VCD files, and the Makefile. Each of your programming tasks **must** be executable with your Makefile or with a standard python interpreter. If your python solution uses non-standard packages you have to provide a requiretmens.txt file. If you do not follow this format, your handing may be graded with 0 points!

Exercise 1 Questions about the lecture [4 Points]

(a) What is a critical path and what effect does it have on the maximum cycle period?

(b) What are the steps involved in a register transfer synthesis? What is meant by elaboration?

(c) What is resource sharing and what conditions must be met for the synthesis tool to perform it automatically?

(d) Why use multi-stage logic for mapping combinational design? What are the issues that need to be solved?

(e) What is a technology-specific netlist?

(f) What is meant by slack time and setup time? How are these determined and what are they used for during synthesis?

Exercise 2 Register Transfer Synthesis [4 Points]

In this exercise, you will look at the general process of register transfer synthesis (RT synthesis) and examine the steps performed by a synthesis tool (in this case Genus from Cadence). For this purpose, we provide you with the finished PLD from exercise sheet 3. The PLD is set to one input and one output. Describe how the netlist changes in the individual steps and classify the steps in the context of the lecture. Also provide a screenshot of the Genus schematic tab for each step. We use the NANGATE standard cell library for this lecture. *Note:* You can exit Genus with the command `exit`.

(a) Execute the command `make synthesis_gui` and wait until the GUI of the synthesis tool has opened. Then add a `Schematic` tab in the middle window, as you can see in Figure 1 and switch to this tab.

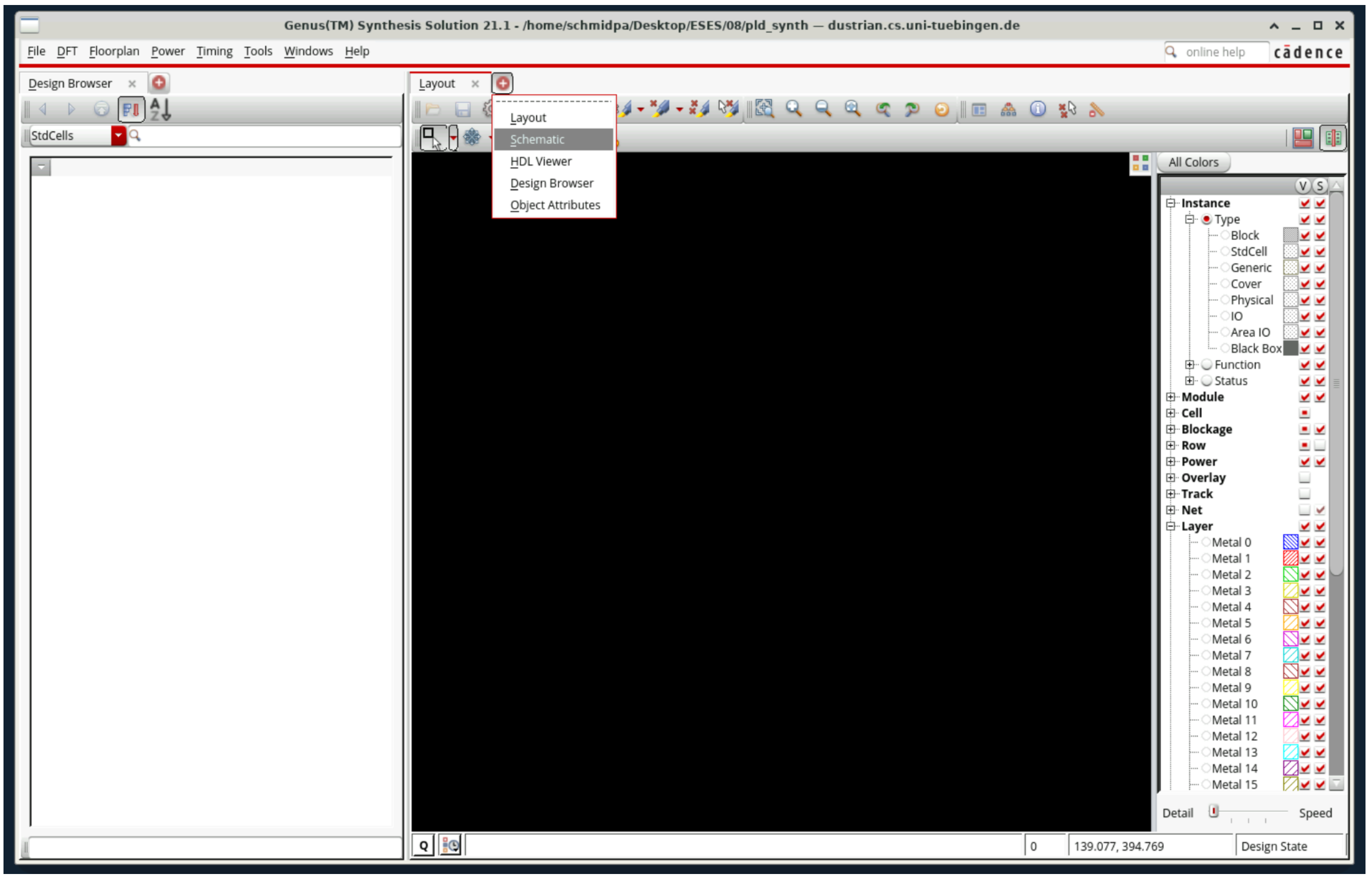

Figure 1: Schematic-Pane Genus

(b) Execute the command `elaborate pld`.

(c) Execute the command `syn_generic`.

(d) Execute the command `syn_map`.

(e) Execute the command `syn_opt`.

Exit Genus and perform a complete synthesis without GUI using `make synthesis`. The synthesis creates a folder named *synthesis* in the same directory as the Makefile. Complete the following tasks:

(a) Now take a look at all report files (file extension .rpt) and describe what you can find in them. Explain what the connection is between the design and the information from the report files.

(b) Look at the post-synthesis netlist in the *synthesis* folder (`syn_pld.v`) and describe what the synthesis has done with the RT netlist.

Hint: After synthesis, you cannot pass generic parameters to the instance in your testbench. You can either remove the parameters when using the post-synthesis design when instantiating it in the testbench, or use an `` `ifdef SYNTHESIS`` (check exercise sheet 5 for how to use it).

Exercise 3 Adder architectures [6 Points]

In exercise sheet 2, you looked at different adder architectures and presented their characteristics analytically. You will find an implementation of these architectures in the subfolder `adder_arch`. The carry lookahead adder is implemented with a fixed bit size of 8 bits. The size of the ripple carry adder can be set with a parameter. Complete the following tasks:

(a) Synthesize both architectures with an input size of 8 bits and compare the results. Pay particular attention to the size of the designs and their time behavior. Comment on whether the result makes sense for you.

(b) Compare the netlist of the carry lookahead adder in the step after the command `syn_map` (see previous task) with the ideal netlist of Wikipeda. Describe how the netlists differ, what you would have expected and how the synthesis tool behaves.

(c) In the file `GATES.txt` you will find a list of all gates within the NANGATE technology. Explain the behavior of the synthesis tool with this new information.

(d) What are the implications of your discovery for other digital designs? What would you like to see in a technology, or what do you think would be helpful? Give reasons for your answers.

Exercise 4 Time behavior of synthesized designs [6 Points]

Use the implementation of the ripple carry adder and design a module as shown in Figure 2. The outputs of the adder are saved in registers. A value should be accepted at a positive clock edge. The reset is `low-active synchronous` and initializes the registers with decimal value 0. You can use a bit size of 8 bits when instantiating the ripple carry adder.

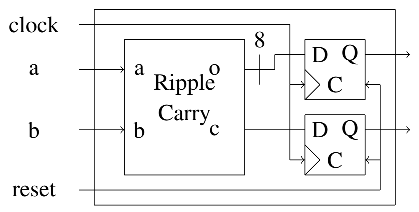

Figure 2: Clocked Ripple Carry

Work on the following tasks after you have implemented the module:

(a) Write a testbench for the design (at least 10 test cases, overflow must also be tested)

(b) Synthesize your design with a cycle time of 1000 ps. Check whether your design achieves this clock period by looking at the generated reports. Repeat the synthesis with a higher clock period if your design does not reach the timing target.

(c) Simulate your design after synthesis with the command `make post_sim` and a clock period higher than the clock period of the synthesis.

(d) Simulate your design after synthesis with the command `make post_sim` and a clock period significantly smaller than the clock period of the synthesis. Describe which warnings occur and why.

(e) Simulate your design, with the small clock period, after synthesis with the graphical user interface and look at the result of the simulation. Describe how an insufficient clock period becomes apparent. Use screenshots of your simulation for your explanation.

CAUTION: Do not forget to enter the name of the instance of your module within the testbench in the variable `HDL_SYN_TOP_TB_INST` in the Makefile! Also fill in every field in the constraints. Use the port names of your top module (not the testbench). Use the forum for questions!