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 7 : Deadline 06.07.2025 - 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!

Introduction:
The following tasks are based on a the graph library from the last exercise sheet. We've extended the file `graph.py` with implementations for the ASAP and ALAP algorithm, as well as functions that you need to complete. For your submission, we expect a short explanation in a PDF how your implementation of the following algorithms works. You must also add the generated graphs from your test cases to the PDF. Submit a ZIP archive containing the PDF and the Python file with your code.

Exercise 1 Binding [10 Points]

In the lecture you learned that, given a schedule, you can perform a computation/function unit assignment using compatibility graphs and clique covering problem. In this assignment, you will implement this algorithm. To implement the algorithm, you need an HLS library that combines a set of operation types into an abstract function unit. The main function of the Python file provided to you contains an example of what an HLS library looks like. You can extend the HLS libraries as you wish. For the sake of simplicity we do not allow that an operation type can be covered by multiple function units.
The following tasks refer to the function `ressource_binding` in the Python file.

(a) Complete the task under the comment `1. Add edges`. To do this, you must connect all nodes in the compatibility graph in the Python variable `comp_graph` that can be executed on a resource (functional unit) without conflict (analog to the definition in the lecture).

(b) Complete the task under the comment `2. Make cliques`. For this, you must find a minimum set of cliques so that that all nodes are covered. *Note:* You can use all networkx functions.

(c) Implement 2 more graphs and verify the correctness of your implementation. Use the ASAP and ALAP algorithm as the base of your schedule.

Exercise 2 Assignment [10 Points]

In this task, you will implement a simple version of the left-edge algorithm (LEA) from the lecture. This algorithm enables you to find the number of registers required for an HLS.
The following tasks refer to the function `register_allocation` in the Python file.

(a) Complete the task under the comment `1. Create intervals based on edges`. To do this, you must create a list of tuples with start and end times from the edges of the schedule.

(b) Complete the task under the comment `2. LEA algorithm`. To do this, you must iterate over the list of tuples and find a set of registers that are necessary for the resource binding. To implement the LEA, you must shift the intervals over each other, analogous to the example in the lecture.

(c) Implement 2 more graphs and verify the correctness of your implementation. Use the ASAP and ALAP algorithm as the basis of your schedule. *Note:* You can use the same examples as in task 1.

Hint: If you have implemented everything correctly, you should find a PDF in your output folder (the folder in which you executed the Python file) in which the schedule (ASAP or ALAP) can be seen, with colored nodes and numbered edges.