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 2 : Deadline 18.05.2025 - 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 [10 Points]

(a) You want to carry out a simple Pareto optimisation using a design space with three dimensions.

$$x = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \in X \subseteq \mathbb{R}^3$$

A design characterisation x stands semantically for

$$\text{(longest path, standard-cell area, } \underbrace{\text{Congestion-Factor}}_{\text{wiring complexity}}\text{)}$$

and is analysed with a characterisation function

$$f : X \to \mathbb{R}^2$$
$$x \mapsto \begin{pmatrix} f_{\text{Area}}(x) = x_2 \cdot \frac{1}{1-x_3} \\ f_{\text{Speed}}(x) = x_1 \cdot \frac{1}{1-x_3} \end{pmatrix}$$

The objective of this function is to punish designs, in terms of area, with a lot of standard cell area and many cables. Also, a design with a long longest path and with a high "wire density" is punished in its speed-characterisitc. Figures 1 and 2 show two different architectures of a 3-bit adder. To which design do you assign a higher speed? Which design has a higher area? Give reasons for your answers (you do not need to do any calculations, text is sufficient).

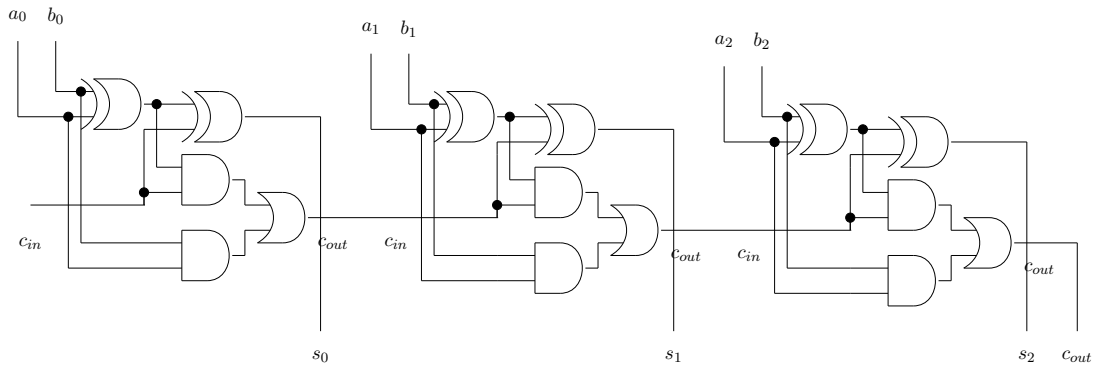

Figure 1: Ripple-Carry-Adder

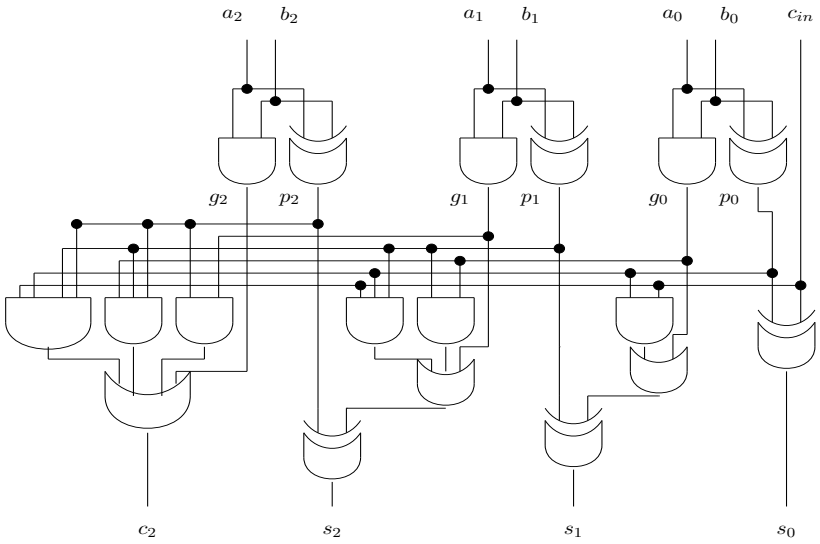

Figure 2: Carry-Lookahead-Adder

Additional Informatoin

A Pareto optimization is used to find an optimal design, given different design parameters. Formally, Pareto optimization is a multi-objective optimization problem in which a design $x^* \in X$ is sought, where $X \subseteq \mathbb{R}^n$ is the set of possible design parameters, which is minimal in respect to an optimization function

$$f : X \to \mathbb{R}^k \quad x \mapsto \begin{pmatrix} f_1(x) & \cdots & f_k(x) \end{pmatrix}^T$$

$$x^* = \min_{x \in X} \left(f_1(x), \ldots, f_k(x) \right).$$

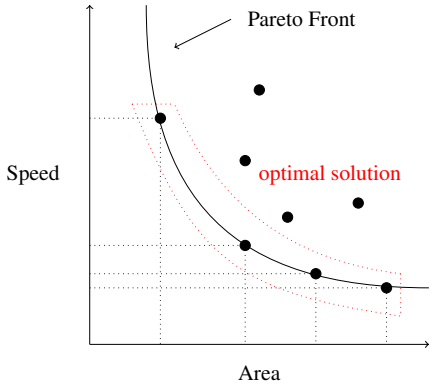

Figure 3: Pareto-Optimization

Normally, at least two criteria are used, i.e. $k \geq 2$, e.g. like in the task. The more criteria are chosen, the more difficult it becomes to solve the optimization problem. A solution is optimal if it is just as good as all other solutions and is better in at least one criterion:

$$\forall x_1, x_2 \in X : x_1 \neq x_2 : \forall i \in \{1, \ldots, k\} \; f_i(x_1) \leq f_i(x_2) : \exists i \in \{1, \ldots, k\} \, f_i(x_1) < f_i(x_2)$$

(b) Model the area of both circuits in figures 1 and 2 with a simple analytical model. Use only the number of logic gates in your model (ignore the "cable density"). Your model receives the word width of the operands as input. In this task, you can assume that the area of individual gates is linear to their fanin (number of input ports). For example, an AND gate with fanin 3 has the size 3AU (Area Unit). Assume the following area as the minimum size for each gate with fanin 2:

AND-/OR-Gate 2AU, XOR-Gate 3AU

(c) Explain how design parameters can be represented in SystemVerilog and name 2 possible applications.

Exercise 2 XOR-Cipher-Block-Chaining [7 Points]

Figure 4 shows the computation of a simple cipher block chaining (CBC) cipher. During encryption, the start chunk (first input chunk) of a message is linked with a key using an XOR operation. The result represents the first output chunk, as well as the key of the next input chunk. Decryption works in the same way.

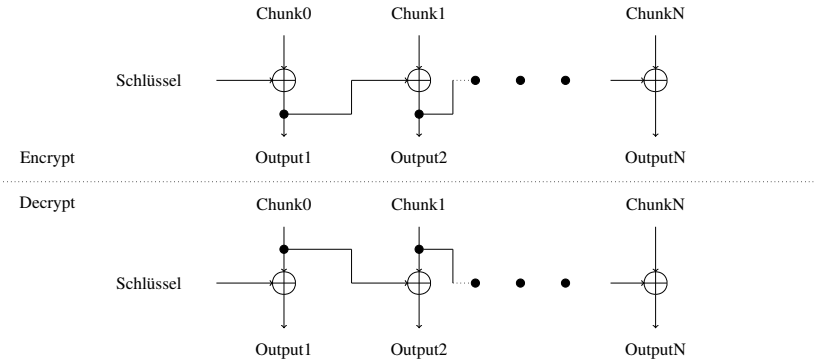

Figure 4: Cipher-Block-Chaining

In this task, a chunk has the size: 8 bit $\cdot n$ and should model an ASCII string. A word/phrase is made up of a series of chunks. The size of a chunk is freely selectable. For example, the following splits are both possible:

$$\overbrace{\text{HE}}^{\text{Chunk0}}\ \overbrace{\text{LL}}^{\text{Chunk1}}\ \overbrace{\text{O}\square}^{\text{Chunk2}}\ \overbrace{\text{WO}}^{\text{Chunk3}}\ \overbrace{\text{RL}}^{\text{Chunk4}}\ \overbrace{\text{D}\square}^{\text{Chunk5}}$$

$$\overbrace{\text{H}}^{\text{Chunk0}}\ \overbrace{\text{E}}^{\text{Chunk1}}\ \ldots$$

The maximum input length therefore is: $8\text{bit} \cdot n \cdot m$, where n is the number of bytes per chunk and m is the number of chunks in the message. The length of the key is equal to a chunk size.

1. Design a component that returns the XOR of a given chunk and a key. Use the base type as the data type of the ports: `input/output logic`. Parameterize the component using parameter n as described above. Design a simple testbench to verify the functionality of your design.

2. Design a second component that works on an entire message. Parameterize the component using parameters n and m as described above. This component should behave like the encryption of the CBC cipher in Figure 4. The component is given a message and a key. Refer to the previous explanation to determine the port bit sizes.

3. Design a testbench with at least 5 test cases with different messages and keys.

4. Modify your CBC component so that it can be set to encryption or decryption mode via an additional port: `input logic enc_dec`.

5. Expand your testbench and show that your encryption and decryption work correctly. You only need to submit the VCD file which shows the encryption and decryption together.

Exercise 3 ALU with Tightly Coupled Encryption/Decryption-Engine [3 Points]

1. Design an ALU as shown in the following figure. Use the provided type declaration for the implementation of your multiplexer (MUX). The `enc_dec` input of the CBC component is implicitly controlled by the selected instruction. The operand inputs and the output should be 16bit.

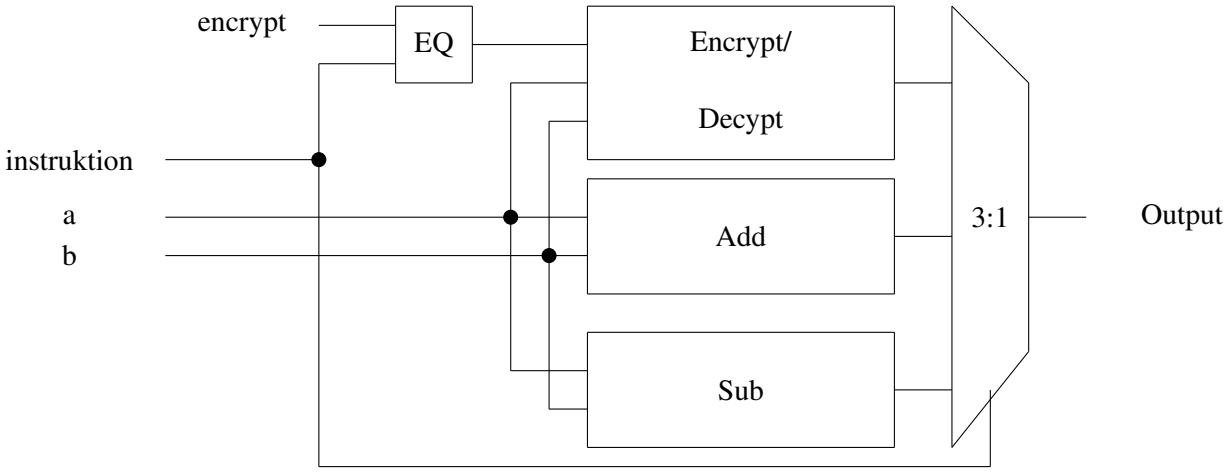

Figure 5: ALU with Tightly Coupled Encryption/Decryption-Engine

2. Design a testbench for the ALU and write 2 testcases for each instruction.

IMPORTANT: It is possible that Ilias has changed the file extension of the source files from `something.sv` to `somethingsv.sec`. Please change the file extension before using the file in your submission. As submission we expect a ZIP archive with two subfolders for the programming tasks. Each subfolder should contain a Makefile for this task.