Computer Engineering – Submission due 17.11.2025

Jan Vogt, Yannik Köllmann - in gleichen Teilen

13. November 2025

1 Arithmetic Logic Unit

1.1 Designing a Basic ALU

1.1.1 Building Blocks

Tasks

1. Implement the module `adder` in the file `adder.sv`. Test your implementation in the testbench `adder_tb` in the file `adder_tb.sv`. Check at least three test cases!

   ```
   (siehe ./src/task5.1/adder/adder.sv)
   ```

   ```
   (siehe ./src/task5.1/adder/adder_tb.sv)
   ```

2. Implement the module `mux_2` in the file `mux_2.sv`. Test your implementation in the testbench `mux_2_tb` in the file `mux_2_tb.sv`. Check at least three test cases!

   ```
   (siehe ./src/task5.1/mux_2/mux_2.sv)
   ```

   ```
   (siehe ./src/task5.1/mux_2/mux_2_tb.sv)
   ```

3. Implement the module `mux_4` in the file `mux_4.sv`. Use the previously implemented module `mux_2` in your implementation of `mux_4`! Test your implementation in the testbench `mux_4_tb` in the file `mux_4_tb.sv`. Check at least three test cases!

   ```
   (siehe ./src/task5.1/mux_4/mux_4.sv)
   ```

   ```
   (siehe ./src/task5.1/mux_4/mux_4_tb.sv)
   ```

1.1.2 Putting the Parts Together

Tasks

1. Fill in the missing parts of Table 5.1.1.

Tabelle 1: Example inputs and outputs for our basic ALU.

i_a	i_b	i_alu_ctrl	o_result	o_carry_out
8’b0000_0000	8’b0000_0000	2’b00	8’b0000_0000	1’b0
8’b1011_1101	8’b1010_0101	2’b00	8’b0110_0010	1’b1
8’b1011_1101	8’b1010_0101	2’b01	8’b0001_1000	1’b1
8’b1011_1101	8’b1010_0101	2’b10	8’b1010_0101	1’b1
8’b1011_1101	8’b1010_0101	2’b11	8’b1011_1101	1’b1

2. Implement the module `alu` in the file `alu.sv`. Test your implementation in the testbench `alu_tb` in the file `alu_tb.sv`. Add respective tests for all examples in Table 5.1.1 to `alu_tb`.

(siehe ./src/task5.1/alu/alu.sv)

(siehe ./src/task5.1/alu/alu_tb.sv)

3. Generate a waveform plot illustrating the application of your ALU w.r.t. Table 5.1.1's inputs. Limit your plot to these inputs and change the input every 10 time units, i.e., visualize 50 time units. The plot shall show all inputs, i.e., `i_a`, `i_b` and `i_alu_ctrl`, and all outputs, i.e., `o_result`, `o_carry_out`.

   (siehe ./src/task5.1/alu/alu_tb_waveform.png)

1.2 Basic ALU in Practice

Tasks

1. Implement the top-level module `alu_de10_lite`. Use the template in Listing 5.2.1.

   (siehe ./src/task5.2/alu_de10_lite.sv)

2. Compile your finished ALU in Quartus Prime and program the FPGA of a DE10-Lite board.

   (siehe ./src/task5.2/quartus_project/)

3. Make sure that the boards shows the correct results for the inputs in Table 5.2.1. Provide a picture of the board for each of the inputs.

   (siehe ./src/task5.2/images/config1.png)

   (siehe ./src/task5.2/images/config2.png)

   (siehe ./src/task5.2/images/config3.png)

   (siehe ./src/task5.2/images/config4.png)

1.3 NZCV-Extended ALU

Tasks

1. Implement the module `xor_3` in the file `xor_3.sv`. Test your implementation in the testbench `xor_3_tb` in the file `xor_3_tb.sv`.

   (siehe ./src/task5.3/xor_3/xor_3.sv)

   (siehe ./src/task5.3/xor_3/xor_3_tb.sv)

2. Fill in the missing parts of Table 5.3.2.

Tabelle 2: Example inputs and outputs for our extended ALU.

i_a	i_b	i_alu_ctrl	o_result	o_nzcv
32'h0000_0000	32'h0000_0000	2'b00	32'h0000_0000	4'b0100
32'h0000_0000	32'hffff_ffff	2'b00	32'hffff_ffff	4'b1000
32'h0000_0001	32'hffff_ffff	2'b00	32'h0000_0000	4'b0110
32'h0000_ffff	32'h0000_0001	2'b00	32'h0001_0000	4'b0000
32'h0000_0000	32'h0000_0000	2'b01	32'h0000_0000	4'b0110
32'h0001_0000	32'h0000_0001	2'b01	32'h0000_ffff	4'b0010
32'hffff_ffff	32'hffff_ffff	2'b10	32'hffff_ffff	4'b1000
32'hffff_ffff	32'h7743_3477	2'b10	32'h7743_3477	4'b0000
32'h0000_0000	32'hffff_ffff	2'b10	32'h0000_0000	4'b0100
32'h0000_0000	32'hffff_ffff	2'b11	32'hffff_ffff	4'b1000

3. Implement the module `alu_nzcv` in the file `alu_nzcv.sv`. Test your implementation in the testbench `alu_nzcv_tb` in the file `alu_nzcv_tb.sv`. Add tests for the examples in Table 5.3.2 to `alu_nzcv_tb`.

   `(siehe ./src/task5.3/alu_nzcv/alu_nzcv.sv)`

   `(siehe ./src/task5.3/alu_nzcv/alu_nzcv_tb.sv)`

4. Generate a waveform plot illustrating the application of your extended ALU to Table 5.3.2's inputs. Apply each input for 10 time units and limit the plot to a total time of 100 time units. Visualize all inputs (`i_a`, `i_b` and `i_alu_ctrl`) and all outputs (`o_result`, `o_nzcv`).

   `(siehe ./src/task5.3/alu_nzcv/alu_nzcv_waveform.png)`