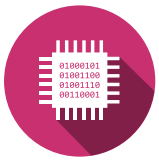


Binary Adders

Labor Digital Design

Contents

1 Goal

This lab aims to learn the design of iterative arithmetic circuits and to implement them practically. The focus is on the development of an adder as a basic arithmetic circuit.

In addition, it is shown how the created adders can be used to realize a subtractor. Through targeted modifications, the concept of binary subtraction is taught, allowing for a comprehensive understanding of arithmetic operations in digital circuits.

2 Carry Ripple Adder

In digital circuits, adders are used to add binary numbers. A basic implementation is the Carry Ripple Adder, which consists of several cascaded adder blocks in which the carry propagates from one position to the next.

2.1 Circuit

A Carry Ripple Adder consists of several iterative blocks, each of which adds two equivalent bits (a_i, b_i) together with an incoming carry (c_i). Each block generates:

- A sum bit s_i, representing the result of the addition.
- An output carry c_{i+1}, which is passed to the next block.

The Figure 1 shows the circuit principle of such a Carry Ripple Adder:

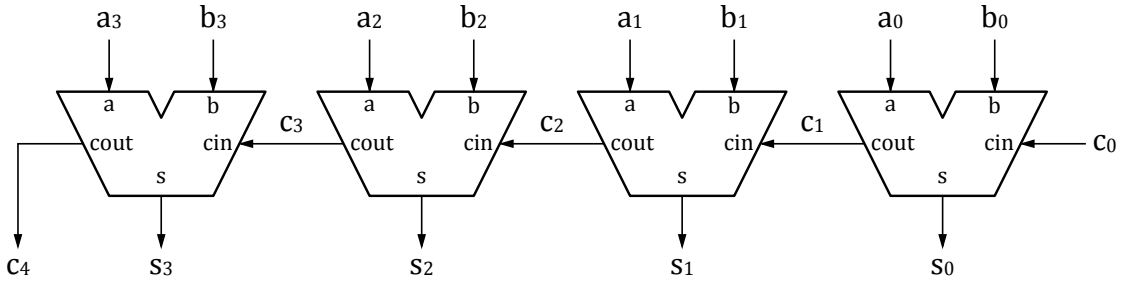


Figure 1 - Carry Ripple Adder

2.2 Implementation

Start by creating the truth table, derive the Boolean equation for the sum bit s_i and the carry c_{i+1}. Finally, develop the combinatorial circuit using INV, AND, OR, XOR gates.

Then implement the iterative scheme of a 4-bit Carry Ripple Adder in the project **`ADD/add4`**.

2.3 Simulation

Every implemented circuit must be correctly tested. The testbench **`ADD_test/add4_tester`** must verify the **full functionality** of the circuit. Answer the following questions:

1. How many test cases are theoretically necessary to verify the correct operation of the adder?
2. Which practical test cases are necessary to verify the correct operation of the adder?
3. How long does the simulation take with the practical test cases?

In the **`add4_tester`**, some *example* test cases are already implemented, you can use them for inspiration and you must adapt them. In Listing 1 you will find such a test case, which checks if the addition $s = a + b + c_{\text{in}} = 0 + 0 + 0 = 0$ is correct. Otherwise, the message **`test 1 wrong`** is displayed in the Modelsim terminal.

```
--------------------------------------------------------------------------------
-- Test 1:    0 + 0 + 0 = 0
--
a   <="0000";
b   <="0000";
cin <='0';
WAIT FOR clockPeriod;
assert (cout = '0') AND (s="0000")
  report "test 1 wrong"
  severity note;
```

Listing 1 - Example test case 1

Complete the **ADD_test/add4_tester** with the necessary tests to verify the full operation of the adder.

Simulate the Testbench **ADD_test/add4_tb** with the simulation file **$SIMULATION_DIR/ADD1.do**.

Investigate and find the longest propagation path in the adder.

3 Subtractor

A subtractor is a circuit that performs the subtraction of two numbers. As we have learned in the course, it is possible to invert the sign of a number in two's complement. This forms the basis of a subtractor.

3.1 Circuit

The circuit of a subtractor can be realized on the basis of the previously developed adder. To perform this subtraction, one can add the complement of the value to be subtracted:

$$a - b = a + (-b) \tag{1}$$

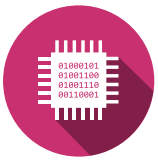
In two's complement, the complement of a number is obtained by inverting all bits of the number and adding 1 to this intermediate result. The inversion of all bits is done with an inverter for each bit. The addition of 1 can be done by acting on the very first carry of the adder **c_0**.

In the block **ADD_Test/sub8_tb**, there are 2 adders of 4 bits each, chained to form an 8-bit adder. There is also a block to invert the bits of the number b.

3.2 Implementation

Modify the circuit to create an 8-bit subtractor.

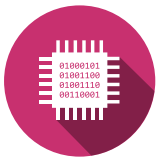
Draw the diagram of the **ADD/forSubtraction** block, which inverts the bits of the number b. Explain why the signal c_{in} is inverted.



3.3 Simulation

Each implemented circuit must be correctly tested.

Simulate the Testbench **`ADD_Test/sub8_tb`** with the simulation file **`$SIMULATION_DIR/ADD3.do`**.



4 Checkout

Before leaving the lab, make sure you have completed the following tasks:

- [ ] Circuit Design
  - [ ] The carry-propagation adder **`ADD/add4`** has been designed and tested.
  - [ ] The subtractor **`ADD_test/sub8_tb`** has been designed and tested.
- [ ] Simulations
  - [ ] The specific tests of the respective testbenches (**`ADD_test/add4_tb`** and **`ADD_test/sub8_tb`**) have been adapted to the circuit.
  - [ ] The simulations contain no errors and all calculations are correct.
- [ ] Documentation and Projectfiles
  - [ ] Ensure all steps (design, circuit, simulations) are well-documented in your lab report.
  - [ ] Save the project to a USB stick or the shared network drive (**`\\filer01.hevs.ch`**).
  - [ ] Share files with your lab partner to ensure work continuity.