Laboration 2

Digitalteknik med VHDL
Adder

Najeem Lawal

ver.:February 4, 2022

1 Adder

This lab-exercise shows how VHDL can be used in different ways to describe the same functionality and how the different descriptions can be verified in the same test bench.

1.1 How to start

Write a synthesizable model fo a 1-bit full-adder(fa) in the developments tool's text-editor. name the file `fa.vhdl`. The module should be like:

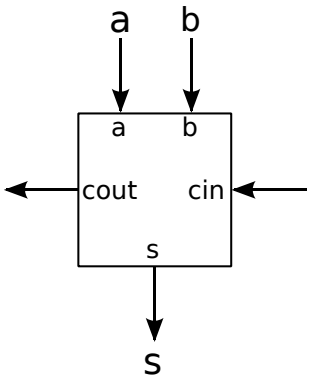


Figure 1: Full adder.

```
entity fa is
    port  (a, b, cin : in std_logic;
              s, cout : out std_logic);
end fa;
```

1.2 Simulate the full adder

Include the file `testfa.vhd` into the development tool's project. This file is supplied to you on the course web-page and describes a test bench. This test bench instantiates a component of your adder and generates a number of input vectors. Run the test bench in your simulator and look at the result between 0 ns and 100 ns. You can zoom

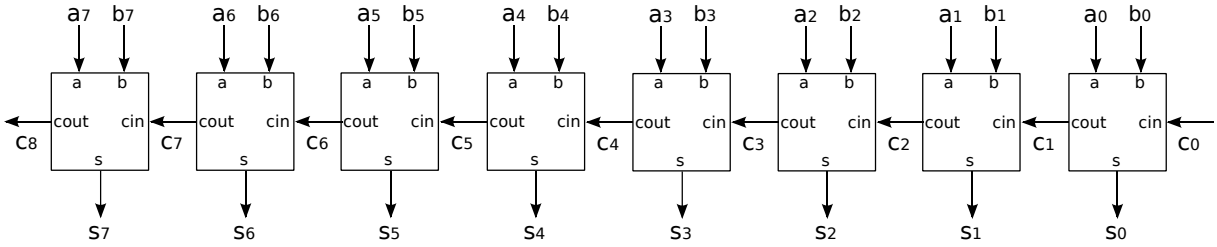
in, zoom out or select a timeframe for your simulator's waveform viewer (ISim) with the buttons in the waveform viewer' s toolbar. Verify manually that your full adder responds correctly to the input stimuli by studying the simulated waveform.

1.3 Test the program on your FPGA board

Check that you synthesized the code for the right FPGA otherwise the programming will fail. Make sure that you have the added a constraints file with the right switches and LEDs with the ports described in your entity. To proceed create the bitstream and transfer it to the device attached to your computer by using the hardware manager. If everything went well, you can now use the physical switches to test your full adder.

2 Structural description of an 8-bit adder

Write a structural description of an 8-bit adder (add8) by using the previously designed full- adder. Name the architecture struct. You will build this 8-bit adder by connecting the eight 1- bit adders such as in the figure below.



The module-interface should be like:

```
entity add8 is
    port ( a, b : in std_logic_vector(7 downto 0);
         cin : in std_logic;
           s : out std_logic_vector(7 downto 0);
        cout : out std_logic);
end add8;
```

2.1 Behavioural description

Write another architecture named *Behave* that implements the same entity description *add8*. This architecture should be written at a higher abstraction level just using the + operator. Both the behavioural and the structural architectures can be written together with the entity description in the same file.

2.2 Simulate the 8-bit adder

Include the file *add8_tb.vhd* into the development tool's project. This file is supplied to you on the course web-page and describes a test bench. The test bench instantiates two components of your 8-bit adder, one component based on the structural description and another component based on the behavioural description. All combinations of input stimuli are generated and the corresponding outputs from both adder components are

compared for equality. Study the test bench code and try to understand how it works. Run a simulation using the supplied test bench and also check the result seen on the waveform output.

2.3 Report

Write a report where you document what you have done and submit a PDF document in Moodle. Other formats are not accepted. Hand in the report latest two weeks after the lab session. This is an individual lab assignment.