You have previously submitted this assignment with David Rama Jimeno. Group can only change between different assignments.

This course has already ended.

« 6. Exercise 4 (/comp.ce.240/spring-2024/... 7. Exercise 5 » (/comp.ce.240/spring-2024/...

COMP.CE.240 (/comp.ce.240/spring-2024/) / 6. Exercise 4 (/comp.ce.240/spring-2024/E04/) / 6.1 Exercise 04: Multi port adder (Structural description)

Assignment description (/comp.ce.240/spring-2024/E04/multi_port_adder/)

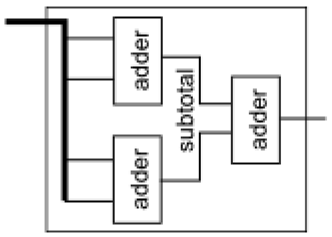
My submissions (1/1000)

Exercise 04: Multi port adder (Structural description)

The purpose of this exercise is to learn:

- How to connect VHDL-blocks together
- Keyword `component`
- How to introduce new signal types (keyword `type`)

The task is to implement a multiport adder using the adder block of the last exercise (https://plus.tuni.fi/comp.ce.240/spring-2024/E03/adder/) (s = a1 + a2 + ...)



Even though the interface of the implemented adder is generic, the basic version needs to support only four inputs

Creating the design

- Create a file with the name `multi_port_adder.vhd`

- Take `ieee`-library and `std_logic_1164`-package into use

- Create an entity with the name `multi_port_adder`
  - Generic parameters with type `integer`
    - `operand_width_g` with default value 16
    - `num_of_operands_g` with default value 4
  - Four ports (3 inputs, 1 output)
    - `clk, std_logic`
    - `rst_n, std_logic`
    - `operands_in, std_logic_vector` with width `operand_width_g*num_of_operands_g`
    - `sum_out, std_logic_vector` with width `operand_width_g`

- Define architecture
  - Name: `structural`
  - Before the `begin` keyword
    - Introduce the `adder` component. There are of course multiple ways to do this:
      - E.g., copy the definition of the entity from `adder.vhd` file and change the syntax by hand or
      - use *Port* > *copy* and *Port* > *paste as component* in (X)Emacs's VHDL-mode (the cursor has to be inside the definition of the entity but the text does not have to be selected) or
      - use the keyword `component` (should start a wizard in Emacs which asks for the names and types etc. (see notes below))
    - Introduce a new type and a signal of this type
      - The type is an array of range `0 ... num_of_operands_g/2-1` of type `std_logic_vector` with width `operand_width_g+1`
      - Signal named `subtotal` of the defined type
      - This creates an array which contains vectors for the subtotals from the first adders
    - In addition, define a signal `total` of type `std_logic_vector` with width `operand_width_g+2`
  - After the keyword `begin`
    - Instantiate the first two adders either
      - by hand or
      - Emacs: use the keyword `instance` (should start Emacs-wizard (see notes below)) or
      - Emacs: copy them with command *Port* > *paste as instance* in the VHDL-mode
    - Use parts of `operands_in` port as inputs for the first adders and connect their outputs to the `subtotal` signal
    - Instantiate a third adder which sums `subtotal(0)` and `subtotal(1)`, together and puts the result to the `total` signal

        - Finally, connect part of the signal `total` to the output `sum_out` leaving two most significant bits unconnected
    - Insert *assert* which ensures that `num_of_operands_g` is always 4 (this is classified as severe failure, i.e., mark "severity failure")
    - Do not use processes

Miscellaneous

- Signal widths require very much caution. Make sure that for all ports, you connect only signals that have the same width as the port.
- Comment your code carefully
- Make sure that your code is indented and aligned for better readability (e.g., automagically Emacs: *VHDL* -> *Beautify* -> *Beautify buffer*)
- Make sure that the block can be compiled without errors (`vcom -check_synthesis`) and loaded to simulator (`vsim multi_port_adder`)
- Test bench for this block is created in the next exercise (https://plus.tuni.fi/comp.ce.240/spring-2024/E05/tb_multi_port_adder/)
- BONUS: instructions here (https://plus.tuni.fi/comp.ce.240/spring-2024/bonus/generic_multi_port_adder/)

Notes about emacs:

- The wizard might not work by default. To fix it, you may need to enable VHDL electric mode first. You can either:
    - From the top bar: *VHDL* -> *Options* -> *Mode* -> *Electric mode* (shortcut *Ctrl-c Ctrl-m Ctrl-e*)
    - Use command *vhdl-electric-mode* in the bottom bar (you may need to press meta+x (meta=alt) first)
- Electric mode has many other wizards as well which can be useful. But if you don't like electric mode, disabling it works the same way as enabling it.
- The other methods for component instantiation etc. told in the guide will work unrelated to state of the electric mode
- More information about the electric mode (http://www.geocities.ws/purnank/vhdlemacst.html)
- See also Emacs quick reference (https://plus.tuni.fi/comp.ce.240/spring-2024/material/emacs-help/)

Return:

Obtain the files specified below and put them under `E04` folder in your Git repository. Remember to do a git push. Return the files also by file return in the bottom of this page.

- Files to return:
    - `answer04.txt`

- Answer with a few sentences to the following questions. Save your answer to file `answer04.txt`.
    - What difficulties could be expected in a generic multiple number adder if the count of numbers is not a power of two or if the count is not pre-defined?
    - What is the difference between VHDL's architecture levels *structural* and *RTL*?
    - How is hierarchical design implemented with VHDL?
- It is enough to answer the questions now as the code will be returned in Exercise 5 (https://plus.tuni.fi/comp.ce.240/spring-2024/E05/tb_multi_port_adder/)

answer04.txt

Choose File No file chosen

Submit with David Rama Jimeno Submit

Earned points

1 / 1

Exercise info

Assignment category
VHDL exercises

Your submissions
1 / 1000

Points required to pass
1

Deadline
Sunday, 11 February 2024, 23:59

Late submission deadline
Friday, 31 May 2024, 23:59

Group size
1-2

Total number of submitters
64

« 6. Exercise 4 (/comp.ce.240/spring-2024/... 7. Exercise 5 » (/comp.ce.240/spring-2024/...