VHDL to Circuit Schematics

The aim of this exercise is to translate VHDL code to circuit schematics. Some VHDL code contains syntax or semantic errors in which case we will say so. Syntax errors include: Missing semicolons and missing `then` or `else` in an if-statement. Semantic errors include: Incomplete sensitivity lists and assigning to the wrong signal. Note that these are just examples of syntax and semantic errors, and there can be other erros in this exercise. We provide the VHDL code for these tasks in a `.zip` file so you can copy it into Vivado to help with finding the syntax errors and correcting the code. **The code shown in this document are segments from the provided VHDL code, only these segments have errors, missing ports and so on are not the errors. You can ignore single `begin` statements with no `end`.**

Hand-in instructions: Prepare a small report with your solutions (detailed). Submit your report as a PDF through the lecture moodle per the moodle submission deadline.

Hints and common errors

- For combinational logic, use `process(all)` like in Listing 1 as it results in much fewer errors when defining a process' sensitivity list. Remember to change the type of your files to VHDL 2008 when using `process(all)`. This is done by selecting the file in the `Source` tab. Look at the `Source File Properties` tab and change the type to `VHDL 2008` by clicking on the 3 dots in the `Type` field.
- Never use `process(all)` for registers! Instead, use `process(CLKxCI, RSTxRI)`.
- You can look at the slides of the previous lectures for help with some of these tasks or just search online for VHDL documentation.
- A good supplement to our teaching material is RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability. This book was not written for VHDL 2008 and will show fully defined sensitivity lists instead of `process(all)`.

Software

To actually write VHDL code, we can recommend the following tools (in-order):

- **Vivado:** As Vivado is already used for this class it is our **first recommendation**. It allows you to use a single tool for all the tasks (writing code, simulating, etc.), however, some of you may find Vivado as a code editor a bit cumbersome.
- **emacs:** emacs (www.gnu.org/software/emacs/) is an extremely extensible editor which is **very powerful for VHDL** and automates several steps of writing VHDL code. We have made a video guide to show how to use this https://mediaspace.epfl.ch/media/11a%2C+Using+EMACS+to+edit+VHDL/0_ytb3o0hz/29738.

If you use something like **Visual Studio Code** there are also various extensions you can install for syntax highlighting.

Task 1

The VHDL code segment in Listing 1 from `task1.vhdl` is an incorrect attempt at describing a 2-1 multiplexer with inputs `In0xDI`, `In1xDI`, `SelxSI`, and output `OutxDO`. Your tasks are to:

1. Describe what is wrong with the code and what would happen if you synthesize the code as it is now. Does it follow our rules for synchronous design? Remember that if you import multiple files and want to see the schematic for just one of them, you have to select that file as the top-level entity before running synthesis. This can be done by right clicking Sources (Hierarchy) window and selecting `Set as top`.
2. Correct the code in two different ways. One solution only needs 1 additional line of code, whereas the other solution needs 2 additional lines of code.
3. Draw a multiplexer symbol along with the signals from the (corrected) code. Indicate inside the multiplexer which signal gets routed to the output depending on `SelxSI`.

Listing 1: VHDL code for Task 1 from `task1.vhdl`.

```
process(all)
begin
  if SelxSI = '1' then
    OutxDO <= In1xDI;
  end if;
end process;
```

Task 2

The VHDL code segment in Listing 2 describes a small piece of hardware. Your tasks are to:

1. There are 2 syntax errors and 1 wrong signal assignment (semantic error). Find and fix these errors.
2. Draw the schematic on paper.

Listing 2: VHDL code for Task 2 from `task2.vhdl`.

```
  signal ResxDN, ResxDP : unsigned(8-1 downto 0);

begin

  process(CLKxCI, RSTxRI)
  begin
    if (RSTxRI = '1') then
      ResxDP <= (others => '0');
    elsif (CLKxCI'event and CLKxCI = '1') then
      ResxDN <= ResxDP;
    end if;
  end process;
  ResxDN <= AxDI + BxDI     when CxDI + DxDI > 1 else
            AxDI - BxDI - 1 when CxDI > DxDI and DxDI /= 0
            AxDI + 1
```

Task 3

The VHDL code segment in Listing 3 describes a small piece of hardware. Your tasks are to:

1. There is 1 semantic error for each process and 2 syntax errors in total. Find and and fix these.
2. Draw the schematic on paper.

Listing 3: VHDL code for Task 3 from `task3.vhdl`.

```
  signal ResxDN, ResxDP : unsigned(8-1 downto 0);

begin

  process(CLKxCI)
  begin
    if (RSTxRI = '1') then
      ResxDP <= (others => '0');
    elsif (CLKxCI'event and CLKxCI = '1') then
      ResxDP <= ResxDN;
    end if;
  end process;

  process()
  begin
    if Sel0xSI = '1'
      ResxDN <= AxDI + BxDI;
    elsif Sel1xSI = '1'
      ResxDN <= AxDI + CxDI;
    else
      ResxDN <= DxDI + 1;
    end if;
  end process;
```

Task 4

The VHDL code segment in Listing 4 is an attempt at describing a free-running binary counter where the signal `MaxPulsexSO` goes HIGH in the same cycle as the counter register `CNTxDP` has the value $(2^4 - 1)_{10} = 1111_2$. However, the pulse is delayed by 1 cycle. Your tasks are to:

1. Explain why the pulse is delayed by 1 cycle and draw a schematic matching the code shown in Listing 4 with no corrections made to the code.
2. This style of writing code is called a *one-segment description* and is considered bad practice! Explain why this style of writing VHDL code is considered bad practice.
3. Fix the code by re-writing it using either a separate process or concurrent assignments for defining the logic for the addition and the pulse. You also need to define additional signals in the code. The code does not have any syntax errors.
4. Draw the circuit schematic for the corrected counter on paper.

Listing 4: VHDL code for Task 4 from `task4.vhdl`.

```
  signal CNTxDP : unsigned(4-1 downto 0);

begin

  process(CLKxCI, RSTxRI)
  begin
    if (RSTxRI = '1') then
      CNTxDP <= (others => '0');
    elsif (CLKxCI'event and CLKxCI = '1') then
      CNTxDP <= CNTxDP + 1;
      if CNTxDP = "1111" then
        MaxPulsexSO <= '1';
      else
        MaxPulsexSO <= '0';
      end if;
    end if;
  end process;
```

Circuit Schematics to VHDL

The aim of this exercise is to translate circuit schematics to VHDL code. When you translate you do not have to write a full module with the ports and everything else, you only have to write the main VHDL code.

Task 5

The circuit schematic in Figure 1 shows an arithmetic circuit where a select signal chooses which result to output. Your task is to translate this circuit schematic into VHDL code.

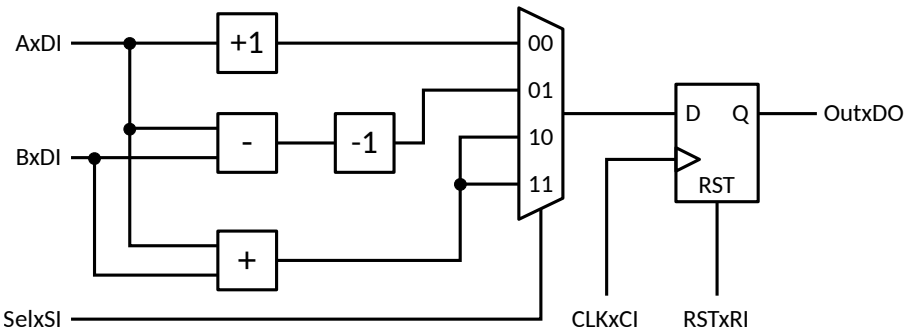

Figure 1: Circuit schematic for Task 5.

Task 6

The circuit schematic in Figure 2 shows a tree structure of comparators with unsigned inputs and multiplexers to route the inputs based on the result of the comparisons. Your tasks are to:

1. Explain what function this block is performing.
2. Translate the schematic into VHDL.
3. Using the provided testbench template file in file `tb_task6.vhdl` as a starting point, try to simulate your design in Vivado. See Lab 1 for how to run a testbench.
4. Modify the testbench to check at least 4 different input combinations to verify the correctness of your design.

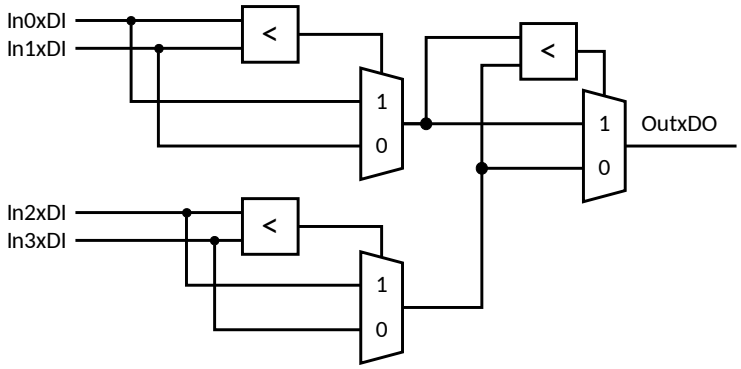

Figure 2: Circuit schematic for Task 6.

Task 7

For the circuit schematic shown in Figure 3, your tasks are to:

1. This circuit is actually quite similar to the one in Task 3. However, there is a significant difference in terms of the hardware you get. Explain what the main difference is between Task 3 and Task 7 in terms of the generated hardware.
2. Translate the schematic into VHDL.

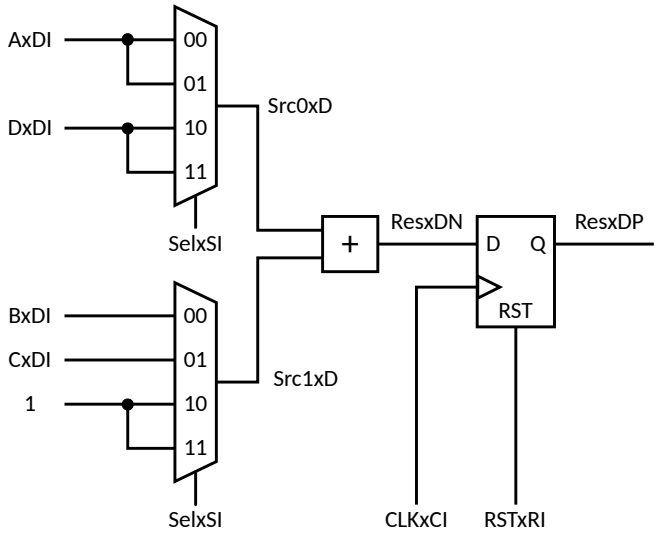

Figure 3: Circuit schematic for Task 7.