FH JOANNEUM
System Test Engineering

Digital Electronics – Laboratory 2

Simulating Combinational Logic

Patrick Lampl

22.10.2020

FH JOANNEUM Gesellschaft mbH
Alte Poststraße 149, A-8020 Graz, AUSTRIA, T.: +43 (0)316 5453-8800, E.: info@fh-joanneum.at, www.fh-joanneum.at

DVR 0813559, UID-Nr ATU 42361001, FN 125888 f, Landesgericht für ZRS Graz
Bankverbindung: Landes-Hypothekenbank Steiermark AG, Kto.Nr. 20141232330, BLZ 56000, IBAN: AT885600020141232330, BIC: HYSTAT2G

1 EXERCISE – SIMULATION WITH VIVADO

Before starting with the Vivado Tutorial, please download the board definition files and extract them to the appropriate folder on your computer: (hint: the board definition files are not required for this exercise but you will need them at some point later during the lectures and you should know how to install them because it is not included within the laboratories standard installation)

https://virtueller-campus-2020-21.fh-joanneum.at/mod/url/view.php?id=14778

Afterwards, work yourself through the Vivado Simulation Tutorial on Moodle, to get to know the Toolchain:

https://virtueller-campus-2020-21.fh-joanneum.at/mod/resource/view.php?id=14779

After this you should have successfully generated a Project, imported some HDL files and executed a simple test bench that instantiated a simple SytemVerilog module called mylogic.sv. Let's summarize what we did there within the HDL code:

- We created a file containing a module that either computes the logic AND or logic OR of two inputs and outputs the result on the "data" port. The input "e" determines which of the two functions is used.
- We created another file which contains our test bench module
    - The test bench module does not have input or output ports! (it doesn't need them)
    - We have instantiated the mylogic module within the test bench module and connected it to signals we also have defined there.
        - We have initialized the three variables: a, b and e to zero (low level)
        - We used a non-synthesizable process (initial block) to create test signals (stimuli) to trigger a reaction from the mylogic module.
        - We used blocking assignments to do the following:
            - After 51ns we switch inputs "a" and "e" high.
                - We should be able to observe in the waveform window that the output is low.
            - After waiting 50ns we switch "b" to high
                - The output should now also go high (a AND b)
            - After another 50ns we switched "b" and "e" to low.
                - Since we now compute "a" OR "b", the output should stay high.
            - After another 50ns we set "a" to zero as well.
                - Since now both inputs to the or gate are zero the output should be zero.
            - Simulation continues for another 200ns before it finishes

2 EXCERSISE – SIMULATION OF LAB 1 CIRCUITS

2.1 Testbench for the comparator

Create a test bench that supplies the three input signals “A” “B” and “output_en” to the “uni_comp” module. Make sure you test all “reasonably” possible scenarios that can appear:

- Make sure you trigger all outputs by suppling all four possible input combinations of “A” and “B”
  - less than “A_lt_B”
  - bigger than “A_bt_B”
  - and equal than “A_eq_B”
- make sure you try to disable the output via setting “output_en” to zero and verify that for all four possible input combinations, all three outputs are definitely zero!

In total this gives you 8 input combinations to test!

Use comments to better remember which condition you are testing at each point in time. Like this it is easier to interpret the results (simulator waveform output)

Simulate the circuit and verify that the circuit behaves as you predict it from its description.

2.1.1 Copy of the source code

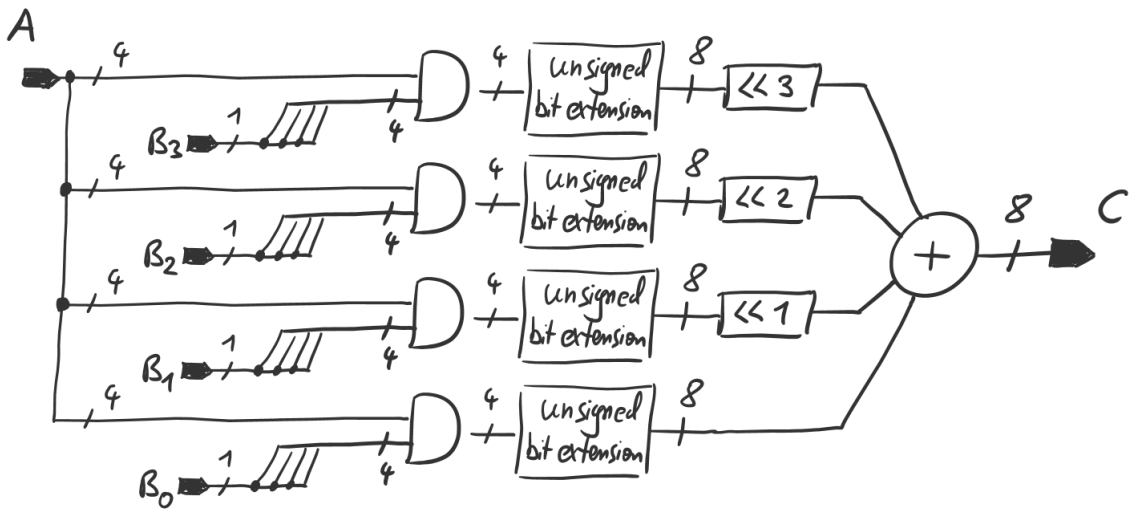
```
1  module uni_comp(
2      input A,
3      input B,
4      input output_en,  //when inactive, outputs are LOW
5      output logic A_lt_B, // A smaller than B
6      output logic A_eq_B, // A equal to B
7      output logic A_bt_B  // A bigger than B
8      );
9
10     always_comb
11     begin
12         if (output_en)
13         begin
14             A_lt_B = (~A) & B;
15             A_bt_B = A & (~B);
16             A_eq_B = ~( A_lt_B | A_bt_B);
17         end
18         else
19         begin
20             A_lt_B = 1'b0;
21             A_bt_B = 1'b0;
22             A_eq_B = 1'b0;
23         end
24     end
25
26 endmodule
```

2.2 Testbench for arithmetic circuit

Create another test bench for this circuit. If I have designed it correctly it should be able to compute the multiplication of the 4-bit inputs “A” and “B”.

Since testing all input combinations of 8 inputs is a little too much for now, we will evaluate only some critical computations to verify the circuit.

- Try a multiplication with arbitrary input values A·B (like 10·5 or 7·3)
- Check if a multiplication with zero always leads to a zero result (test A·0 and 0·B and 0·0)
- Check if a multiplication with the two maximum 4-bit values works (15x15)



Question: does the circuit give valid results if you interpret the inputs to be signed values?
Eg. Try suppling (-2) · 4 which would be 1110 · 0100 in binary.

3 DOCUMENTATION

Create a short report of your results from 2.1 (comparator) and 2.2 (multiplier) and explain whether your simulation results proof or disprove the correct operation of the two circuits modules.

Answer the question raised within 2.2 about the signed data capability of the module

Upload your report to Moodle

4 HINTS AND TRICKS

Here some methods for displaying some text or some results within the console.
You can also see here how to do a for loop so that coding complex stimuli becomes easier.

$display for displaying text such as:

```
$display ("---------- TEST DECODER ----------.");
```

$display for displaying text and variables such as:

```
$display ("content of 8-bit register: %8b", nameofregister);
```

For loops for sequentially testing single elements or all possible input combination of something

```
for (int i = 0; i < 8; i++) begin
            #50ns; //delay
            ...    //code to test a single one of eight elements
            ...    //or a single combination of eight possible combs.
end
```

4.1 Installing Vivado on your private Computer

If you would like to use Vivado on your private computer (which I would highly recommend, so that you can finish your lab assignments outside the classroom), you can download it from the following one drive folder:

https://fhjoanneum-my.sharepoint.com/:f:/g/personal/patrick_lampl_fh-joanneum_at/EtCbnuP5w-5Cl9CKP4sCtXQBKlRAcQVlnKQi_rIvNU0WLg?e=byNWAP

After extracting the zip file you find two readme files that explain how you can create a desktop icon for the program.

The download is 8GB and after extraction it will consume roughly 16GB of disc space.
(this is already the minimum installation possible for your requirements. If you download the original installer you will need to create a Xilinx account and the download itself is already 22GB.)