Introduction to Electronic System Design

Laboratory assignment 3

Test benches and design synthesis

Sven Knutsson
svenk@chalmers.se
Dept. Of Computer Science and Engineering
Chalmers University of Technology
Gothenburg
Sweden

Goals

- Introduce metods to test your VHDL designs
- Test if the code you have written is synthesizable

  Can the code be transfered to hardware?

  Does the synthesized hardware behave
  like the VHDL code did?

Testing the design

We have been using QuestaSim to test our designs

In this we have written script files, do files, for the test

These files can only create the instimuli for the test

To check the correctness of the design we have to manually inspect the resulting output signals

This won´t work for larger designs

We need ways to check the results against the expected outcome

This is where test benches come into play

You can find a desctription of test benches in part 2 of the VHDL presentation

Test benches

Assignment

You should write a test bench of type 3 for the counter specified below

When you write the test bench there is a big risk that you do the same logical misstake as you might have done at design time

To avoid this someone else should write the test bench for your design

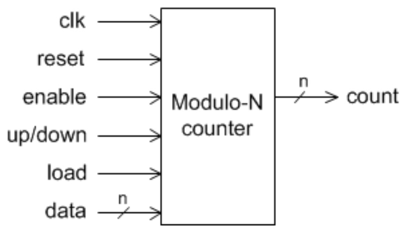
We do this like a real case where we start writing the test bench before the design is finished

This means that you have to write the test bench from reading the specifications for the design

Test benches

Design specification

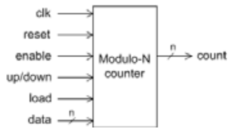
Write a test bench type 3 for a modulo-20 up/down counter



The counter should count from 0 to 19 and then restart (up) from zero or count from 19 down to 0 and then restart (down) from 19

It should have the following features

Test benches



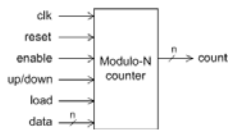
Design specification

- The counter counts clock pulses
- 1 A reset signal sets the count value to zero ← Asynchronous
- 3 An enable signal must be activated for the counting to take place ← Synchronous
- 2 A load signal loads a new value (data) into the counter in parallel. Values outside of the counting range should be ignored
- An up/down signal controls the counting direction
- Reset and load can take place irrespective of the setting of the enable signal

Order of dominance

Test benches

Design specification cont.



The design has the entity

```
ENTITY counter IS
   GENERIC(N:POSITIVE:=20);
   PORT(clk:IN STD_LOGIC;
        reset:IN STD_LOGIC;
        enable:IN STD_LOGIC;
        load:IN STD_LOGIC;
        up_down:IN STD_LOGIC;
        data:STD_LOGIC_VECTOR(INTEGER
                     (CEIL(LOG2(REAL(N))))-1 DOWNTO 0);
        count:OUT STD_LOGIC_VECTOR(INTEGER
                     (CEIL(LOG2(REAL(N))))-1 DOWNTO 0));
END counter;
```

The word length used is set as needed to hold values up to N-1

Note how the bit range of the vectors is calculated. It uses a library called `math_real`

Test benches

Assignment test

How do we find out if the test bench works and catches any errors?

In the lab the assistants will test it on our supplied counter designs.

The supplied designs are given as encrypted files witch means that you can´t read the code.

There is a papers on the homepage on how to simulate encrypted files.

Some of the designs have errors, some don´t.

Try to identify if there are any errors and what might have caused these errors.

Synthesis

Synthesis means turning our code into hardware.

In our case this means placing the design in a FPGA.

Not all code is synthesizable, that is it might not be possible to transfer it to hardware

Here we will synthesize the code and make sure that it is synthesizable

And we will also make sure that the synthesized design works as expected by simulating it after syntesis and implementation

Synthesis

We will use the tool Vivado from Xilinx to synthesize and implement our design

There is a paper on using Vivado on the homepage.

The paper includs descriptions on how to use QuestaSim to simulate synthesized and placed and routed code.

Synthesis

Assignment

Synthesize the designs from lab assignment 1 and 2

- Ripple carry adder/subtracter with overflow and saturation
- Serial adder/subtracter

Compare size and speed

- Direct implemented FIR filter
- Serially implemented FIR filter

Compare size and speed

If the code doesn´t pass the synthesis process try to find out why and rewrite the code.

Not only errors but also warnings from the synthesis process might be important

The same applies if the synthesised code doesn´t pass the simulation

Synthesis

Assignment cont.

Although your code passes the syntesize stage it may still not work the way it should.

The way you write your code may make the synthesis tool get it wrong.

To check this we will simulate the code after it has been placed and routed into the FPGA.