Lab-exercise

Lab6:
Video timing simulation

Introduction

The verification of hardware by means of simulation is an important step in the integrated circuit design process. For ASICs this is more apparent because the high production cost demands a first-time-right design. Whereas for FPGAs this cost is not there, simulation is many times the only way to get insight in the correct functionality of the system because you can observe internal nodes. In practice for each design a set of VHDL testbenches has to be written to functionally verify the design. These testbenches use the whole VHDL language, whereas the synthesizable descriptions only use a subset.

Goals

The goal of this module is making sure the student fully verifies the video timing generator of module1a and, if necessary, debugs it.

- This should happen using the testbench procedures created and tested in module1b.
- The testbenches should be self-checking as much as possible. In practice testbenches are rerun tens or hundreds of times before a system is functionally correct. It is not practical to check all signals graphically after each simulation. Ideally the testbench should be written such that if it reports no errors, the simulation results are correct.

Background knowledge

Basic knowledge of VHLD and Modelsim
The specification for the signals used in driving a VHDL screen [module1a].

Classification

Difficulty level on a scale of 1 to 5 : 4
Time needed (without support): 4 hours

Input

- VideoTimingGen.vhd : Synthesizeable VHDL of the video timing generator [module1a]
- Constants_pack.vhd : package holding (S)VGA timing constants
- General_TB_pack.vhd : package holding testbench procedures en functions [module1b]

The lab

Create a testbench TB_VideoTimingGen in the file TB_VideoTimingGen.vhd for the video timing generator. Use TB_TestHardware.vhd as guideline. If

VideoTimingGen is compiled in library WORK, you can use the following statements to start your testbench.

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
                    LIBRARY Const_lib;
use Const_lib.Constants_pack.all;
LIBRARY General_TB_lib;
use General_TB_lib.General_TB_pack.all;
ENTITY TB_VideoTimingGen IS
END TB_VideoTimingGen;
ARCHITECTURE behav OF TB_VideoTimingGen IS
```

In the architecture place the following VHDL processes:

- Instantiation of the VideoTimingGen component
- Generating the inputs
- Call the GenClock procedure concurrently to generate the clock The pixel klok frequency here is 25 MHz.
- a "main" process generating Clk_en and Reset_n and waiting 2 million clock periods before making the EndOfSim signal true. In this lab Clk_en is not going to be used and should be held '1' continuously.
- Checking the outputs
- Use the CheckPulse procedure to check the high as well as the low level of Hsync and Vsync.
- Also use this procedure to verify Video_on and Start_video
- Use CheckDist to measure the distance between the rising edge of Vsync and a change on Start_Video
- Do the same with Vsync and the Video_on signal, as well as with Hsync and Video_on.
- Use the CheckIfValueOccursInSim to perform some spot checks on the Hcount and Vcount signals.

Start modelsim

- `cd <your module1c folder>`
- `vlib work`
- `vlib Const_lib`
- `vlib General_TB_lib`

- `vmap Const_lib Const_lib`
- `vmap General_TB_lib General_TB_lib`
- `vcom -work General_TB_lib General_TB_pack.vhd`
- `vcom -work Const_lib Constants_pack.vhd`
- `vcom VideoTimingGen.vhd`
- `vcom TB_VideoTimingGen.vhd`
- `vsim work.TB_VideoTimingGen`
- `add wave *`
- `run -all`

Only in the beginning of the simulation (at 0 ns) you should get the following type of messages.

** Warning: NUMERIC_STD.">=": metavalue detected, returning FALSE
Time: 0 ps Iteration: 0 Instance: /tb_videotiminggen/inst
** Warning: NUMERIC_STD.">=": metavalue detected, returning FALSE
Time: 0 ps Iteration: 0 Instance: /tb_videotiminggen/inst

….

These messages are generated when a conditional operator is used on an object of signed or unsigned type. The code below gives an example. At time zero, some_signed_interval is equal to its initialization value “UUUU” . The “<” function in the IEEE.numeric_std package reports the above message to the screen and returns a value as if the value would have been “0000”.

```
process(some_signed_internal) -- signed(3 downto 0)
begin
if (some_signed_internal < 5) then
```

….

The simulation should stop automatically after 2 000 000 clock periods and should not generate any additional messages.