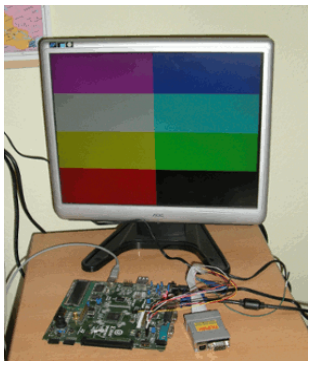
GHDL simulate VHDL code with included Xilinx Library Unisim

René Doß
http://www.dossmatik.de

January 25, 2010

GHDL is a free simulator for VHDL. This tool grows up with a performance of huge VHDL features. It is possible to detect early enough mistakes in design. Xilinx has a manufacture specific library called Unisim for device specific components. This pager show you the way, how it is possible to simulate the Unisim inside your design with GHDL. It is a short instruction. A VGA monitor example is used for a better understanding. The most development boards have a VGA plug. After the simulation you can test this example in hardware. The example starts without unisim components. It has a VGA resolution 640*480 and a pixel clock of 25MHz. It is explained, how you can operate the command line of GHDL. After the example is increased the resolution to 1024*768 and a pixel clock of 75MHz. The clock is multiplied with the DCM.



This three programs have to useable on your PC.

GHDL http://ghdl.free.fr/
ISE 11.4 http://www.xilinx.com/
Gtkwave http://gtkwave.sourceforge.net/

1 VGA-Port Resolution 640x480

The 50MHz input clock is divided by 2 with a Flip-Flop. This is the pixel clock. A column counter runs from 0 to 799. A carry increases the row counter. From these two counter are derived the synchronisation signals and a test picture are generated on the colour channels.

VHDL-Code

The following code give on a VGA monitor a striped pattern. It should be stored with file name vga640_480.vhd.

```
----------------------------------
-- Dossmatik GmbH
-- http://www.dossmatik.de
--simple VGA example

-- Additional Comments:
--
--Spartan3AN board
-- Pin Assignment:
-- NET clk50_in loc = T9
-- NET red_out LOC=R12;
-- NET green_out LOC=T12;
-- NET blue_out LOC=R11;
-- NET hs_out LOC=R9;
-- NET vs_out LOC=T10;
----------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.ALL;

entity vga_timing is
port(
clk50_in : in std_logic;

red_out   : out std_logic;
green_out : out std_logic;
blue_out  : out std_logic;
hs_out : out std_logic;
vs_out : out std_logic);
end vga_timing;

architecture behavioral of vga_timing is
```

```
signal clk25 : std_logic:='0';
signal hcounter : unsigned (9 downto 0):=to_unsigned(0,10);
signal vcounter : unsigned (9 downto 0):=to_unsigned(0,10);

begin

-- generate a 25Mhz clock
process (clk50_in)
begin
if clk50_in'event and clk50_in='1' then
    clk25 <= not clk25;
end if;
end process;

process (clk25)
begin
if clk25'event and clk25 = '1' then
        if vcounter<480 then
            if hcounter < 320 then
                  red_out<='1';
             else
                red_out<='0';
            end if;
        else
            red_out<='0';
        end if;
end if;
end process;


process (clk25)
begin
if clk25'event and clk25 = '1' then
        if hcounter<640 then
            if vcounter < 240 then
                  blue_out<='1';
            else
                  blue_out<='0';
            end if;
            if vcounter > 120 and vcounter < 360 then
                  green_out<='1';
            else
                  green_out<='0';
            end if;
        else
```

```
            blue_out <='0';
            green_out <='0';
        end if;
end if;
end process;


--sync signal generation
process (clk25)
begin

if clk25'event and clk25 = '1' then

        if hcounter >= (639+16) and hcounter <= (639+16+96) then
                hs_out <= '0';
        else
                hs_out <= '1';
        end if;
        if vcounter >= (479+10) and vcounter <= (479+10+2) then
                vs_out <= '0';
        else
                vs_out <= '1';
        end if;
-- horizontal counts from 0 to 799
        hcounter <= hcounter+1;
        if hcounter = 799 then
                vcounter <= vcounter+1;
                hcounter <= to_unsigned(0,10);
        end if;
-- vertical counts from 0 to 524
        if vcounter = 524 then
                vcounter <= to_unsigned(0,10);
        end if;
end if;
end process;

end behavioral;
```

Testbench

The convenient testbench delivers a stimulation of the signal clk50_in. It is the clock for driving all counters.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY tb_vga IS
END tb_vga;

ARCHITECTURE behavior OF tb_vga IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT vga_timing
    PORT(
         clk50_in  : IN  std_logic;
         red_out   : OUT  std_logic;
         green_out : OUT  std_logic;
         blue_out  : OUT  std_logic;
         hs_out    : OUT  std_logic;
         vs_out    : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal clk50_in : std_logic := '0';

 --Outputs

   signal red_out : std_logic;
   signal green_out : std_logic;
   signal blue_out : std_logic;
   signal hs_out : std_logic;
   signal vs_out : std_logic;

   -- Clock period definitions
   constant clk50_in_period : time := 10 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: vga_timing PORT MAP (
          clk50_in => clk50_in,
          red_out => red_out,
          green_out => green_out,
          blue_out => blue_out,
```

```
        hs_out => hs_out,
        vs_out => vs_out
    );

  -- Clock process definitions
  clk50_in_process :process
  begin
            clk50_in <= '0';
            wait for clk50_in_period/2;
            clk50_in <= '1';
            wait for clk50_in_period/2;
  end process;


END;
```

GHDL is a command line tool. The simulation is controlled with Options. First each file have to analysed. If inside the VHDL code is a mistake, GHDL reports a message with a localisation. The option for analyse is -a.

```
ghdl -a vga640_480.vhd
```

```
ghdl -a tb_vga.vhd
```

After an executable file have to build. It is called with the option -e and the top entity port.

```
ghdl -e tb_vga
```

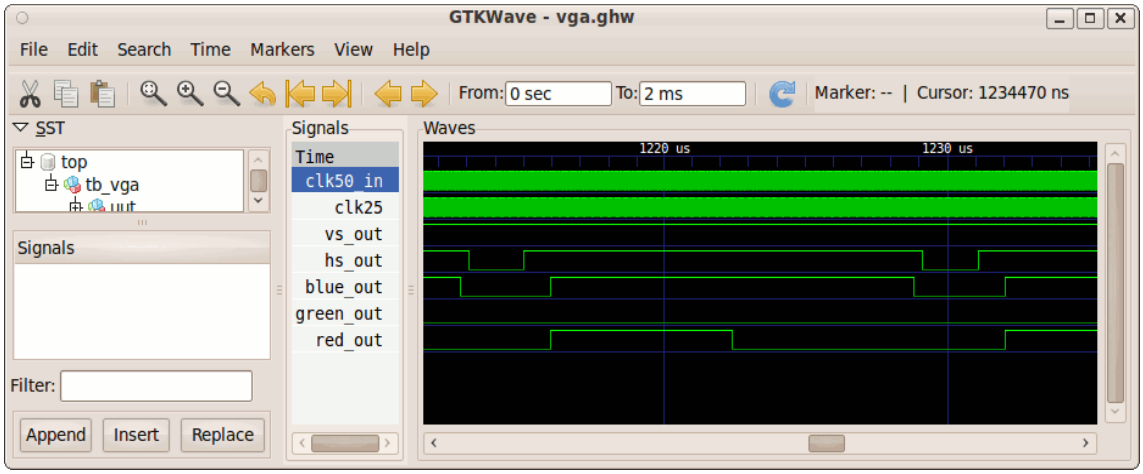
The executable file behaviours like the components are wired, how is written in VHDL Code. The simulation runs for a duration time. The output is written in the output file vga.ghw .

```
ghdl -r tb_vga --stop-time=2000000ns -wave=vga.ghw
```

The simulate signals switching are now stored in the file vga.ghw. The program gtkwave makes viewable the devolution in a time diagram.

```
gtkwave vga.ghw
```

Considered signals have to chosen from the design and to activated the trace. Now the signal waveform is visible inside the time diagram.



2 VGA Port at resolution 1024x768

Now it is not so simple to generate a 75MHz pixel clock. The 50MHz input clock is lower as necessary. The DCM (Digital Clock Manager) is a component from the Unisim Library. It can divide and multiply clock signal. Also some special buffers are used for the routing. The following code change the colour at your monitor. File is stored with the name vga1024_768.vhd. The same testbench is used again for the simulation.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;


entity vga_timing is
port(
    clk50_in     : in std_logic;
    red_out      : out std_logic;
    green_out    : out std_logic;
    blue_out     : out std_logic;
```

```
    hs_out      : out std_logic;
    vs_out      : out std_logic);
end vga_timing;

architecture behavioral of vga_timing is

signal clk75            : std_logic;
signal CLK0_BUF         : std_logic;
signal CLKFB_IN         : std_logic;
signal hcounter         : integer range 0 to 1328;
signal vcounter         : integer range 0 to 806;
signal color            : std_logic_vector(2 downto 0):="111";
signal CLKIN_IBUFG      : std_logic;
signal CLKFX_BUF        : std_logic;
signal GND_BIT          : std_logic;


begin

   GND_BIT <= '0';

   CLKIN_IBUFG_INST : IBUFG
      port map (I=>CLK50_in,
                O=>CLKIN_IBUFG);

   CLK0_BUFG_INST : BUFG
      port map (I=>CLK0_BUF,
                O=>CLKFB_IN);



   CLKFX_BUFG_INST : BUFG
      port map (I=>CLKFX_BUF,
                O=>CLK75);


  DCM_SP_INST : DCM_SP
   generic map( CLK_FEEDBACK => "1X",
            CLKDV_DIVIDE => 2.0,
            CLKFX_DIVIDE => 2,
            CLKFX_MULTIPLY => 3,
            CLKIN_DIVIDE_BY_2 => FALSE,
            CLKIN_PERIOD => 20.000,
            CLKOUT_PHASE_SHIFT => "NONE",
            DESKEW_ADJUST => "SYSTEM_SYNCHRONOUS",
```

```
            DFS_FREQUENCY_MODE => "LOW",
            DLL_FREQUENCY_MODE => "LOW",
            DUTY_CYCLE_CORRECTION => TRUE,
            FACTORY_JF => x"C080",
            PHASE_SHIFT => 0,
            STARTUP_WAIT => FALSE)
        port map (CLKFB=>CLKFB_IN,
                  CLKIN=>CLKIN_IBUFG,
                  DSSEN=>GND_BIT,
                  PSCLK=>GND_BIT,
                  PSEN=>GND_BIT,
                  PSINCDEC=>GND_BIT,
                  RST=>'0',
                  CLKDV=>open,
                  CLKFX=>CLKFX_BUF,
                  CLKFX180=>open,
                  CLK0=>CLK0_BUF,
                  CLK2X=>open,
                  CLK2X180=>open,
                  CLK90=>open,
                  CLK180=>open,
                  CLK270=>open,
                  LOCKED=>open,
                  PSDONE=>open,
                  STATUS=>open);
```

```
-- change color
p1: process (clk75)
variable cnt: integer;
begin
if clk75'event and clk75='1' then
cnt := cnt + 1;
if cnt = 25000000 then
color <= color + "001";
cnt := 0;
end if;
end if;
end process;

p2: process (clk75, hcounter, vcounter)
variable x: integer range 0 to 2000:=0;
variable y: integer range 0 to 2000:=0;
```

```
begin
x := hcounter ;
y := vcounter ;
if clk75'event and clk75 = '1' then
    if x < 1023 and y < 767 then
      red_out <= color(0);
      green_out <= color(1);
      blue_out <= color(2);
    else
    -- if not traced, set it to "black" color
    red_out <= '0';
    green_out <= '0';
    blue_out <= '0';
    end if;

    if hcounter > 1047 and hcounter < 1185 then
        hs_out <= '0';
    else
        hs_out <= '1';
    end if;

    if vcounter > 770 and vcounter < 778 then
        vs_out <= '0';
    else
        vs_out <= '1';
    end if;
        hcounter <= hcounter+1;
    if hcounter = 1238 then
        vcounter <= vcounter+1;
        hcounter <= 0;
    end if;
--
    if vcounter = 806 then
        vcounter <= 0;
    end if;
end if;
end process;

end behavioral;
```

At larger design, GHDL can manage the file construct. It exists dependency in the VHDL structure. The correct sequence is wanted in analysing the files. The automatic feature is to import the files first. The option -i stands for import.

```
ghdl -i --work=unisim /opt/Xilinx/11.1/ISE/vhdl/src/unisims/*.vhd
ghdl -i --work=unisim /opt/Xilinx/11.1/ISE/vhdl/src/unisims/primitive/*.vhd
```

```
ghdl -i *.vhd
```

After the import is to generated the executable file by GHDL. This step is make with option -m.

```
ghdl -m -g -Punisim --warn-unused --ieee=synopsys tb_vga
```

Now it is the simulation like the first example. The simulation runs for a time period. The output is viewable gtkwave.

```
 ghdl -r tb_vga  --disp-tree=inst --stop-time=20000ns --wave=vga.ghw
tb_vga [entity]
‘-behavior [arch]
  ‘-uut [instance]
    ‘-vga_timing [entity]
      ‘-behavioral [arch]
        +-clkin_ibufg_inst [instance]
        | ‘-ibufg [entity]
        |   ‘-ibufg_v [arch]
        +-clk0_bufg_inst [instance]
        | ‘-bufg [entity]
        |   ‘-bufg_v [arch]

......

./tb_vga:info: simulation stopped by --stop-time

 gtkwave vga.ghw
```