VGA Driver Testbench Documentation

17-04-2023

Tags: #SEM4_PROJECT #Version_1
Related files/topics:

- SEM4_PROJECT
- Zynq-7000, VHDL and Vivado
- VGA Driver

Testbench description

The simulation/testbench configuration is fairly straightforward and follows this pipeline:

1. **VGA component port description:*

```
component VGA
      port(
         clk: in std_logic;                           -- 100MHz clock
         RST: in std_logic;                           -- Universal reset
         R_switch, G_switch, B_switch: in std_logic;

         Hsync, Vsync: out std_logic;                 -- Horizontal and vertical sync
         RGB: out std_logic_vector(23 downto 0);      -- Red, Green and Blue outputs in one 24
bit stream
         videoOn: out std_logic);                     -- Indicates if hPos and vPos are in
drawable area
end component;
```

2. **Clock divider component port description:**
   Note: Technically the clock divider entity parts are not needed in the simulation, but they help form a better overview

```
component clk_div
      port(
         Clk_in: std_logic;
         reset: in std_logic;
         CLK_out: out std_logic);
end component;
```

3. **Signal definitions for both VGA and clock divider:**
   In addition to signals a constant for the simulated 100MHz clock period is needed

```
--Inputs for VGA entity
    signal clk: std_logic := '0';
    signal RST: std_logic := '0';
    signal R_switch: std_logic := '1';
    signal G_switch: std_logic := '0';
    signal B_switch: std_logic := '1';

    --Outputs for VGA entity
    signal Hsync, Vsync: std_logic := '0';
    signal RGB: std_logic_vector(23 downto 0);
    signal videoOn: std_logic;

    --25MHz clock
    signal clk_25: std_logic := '1';
```

```
    --100MHz clock period
    constant clock_period: time := 10ns;
```

4. **Port-mapping for the clock divider and VGA driver:**

```
clock_100_25: clk_div PORT MAP(
      Clk_in => clk, reset => RST, CLK_out => clk_25);

uut: VGA PORT MAP(
      clk => clk, RST => RST, R_switch => R_switch, G_switch => G_switch,
              B_switch => B_switch, Hsync => Hsync, Vsync => Vsync,
              videoOn => videoOn, RGB => RGB);
```

5. **Simulating a 100MHz clock source:**

```
--Simulate a 100MHz clock
    clock_process: process
    begin
        clk <= '0';
        wait for clock_period/2;
        clk <= '1';
        wait for clock_period/2;
    end process;
```

6. **Writing RGB values in to a text document:**
   Note: In order to write text into a txt document these need to be imported "use std.textio.all" and "use ieee.std_logic_textio.all";

The process will create a text document names 'rgb' and will write in the RGB values corresponding to where it is in the horizontal line, every 25MHz clock rise. The process will give an error message that will stop the simulation when a full frame has been written, when Vsync will go to 0.

```
--write RGB values in a text document
    record_values:process(clk_25) is
        file     DISP_FILE : text open write_mode is "rgb.txt";
        variable DISP_LINE : line;
        variable h : std_logic := '0';
    begin
        if rising_edge(clk_25) then
            if videoOn = '1' then
                write(DISP_LINE, to_integer(unsigned(RGB)));
                write(DISP_LINE, ',');
                h := '1';
            end if;

            if hsync = '0' and h = '1' then
                h := '0';
                writeline(DISP_FILE, DISP_LINE);
            end if;

            if vsync = '0' then
                assert false report "completed...ignore following error messages" severity
FAILURE;
            end if;
        end if;
    end process;
```