Designing with programmable logic devices - course at the University of West Bohemia

MARTIN POUPA
Department of Applied Electronics and Telecommunications
University of West Bohemia in Pilsen
Univerzitní 26, Pilsen 306 14
CZECH REPUBLIC
poupa@kae.zcu.cz http://home.zcu.cz/~poupa

Abstract: - This article deals with the evolution of teaching of programmable logic devices at the Faculty of Electrical Engineering in Pilsen. Several ways how to verify digital designs in the laboratory exercises are described and finally demonstration example of a simple digital circuit (generator of Fibonacci numbers) with the testbench used in laboratory exercises are shown.

Key-Words: teaching, programmable logic devices, VHDL, computer aided design

1 Introduction

The programmable logic devices (PLD) have been taught for a long time at the University of West Bohemia, Faculty of Electrical Engineering. In the early nineties, programmable logic devices were taught in the extent of several hours of lectures and laboratory exercises. Since the summer semester 2001, the PLDs have been taught in the special facultative subject named "Programmable Logic Devices".

2 Description of subject

In the subject "Programmable Logic Devices" the students are familiarized with the elementals of programming and the utilization of the programmable logic devices, and the methods of the design of combinatorial and sequential digital systems are explained and exercised. For the description of the digital systems, the high-level hardware description language VHDL is taught and used. The VHDL language is taught in accordance with IEEE Standard VHDL Language Reference Manual (IEEE Std 1076-1993 Edition).

There are three architectures of programmable logic devices – the SPLD, CPLD and FPGA. The lectures provide information on the two main architectures (CPLD and FPGA) that is common also to the wide range of PLD devices. Detailed description of CPLD family XC9500 from Xilinx and FPGA family Cyclone from Altera is given to the students. Both mentioned families have favourable price/performance rate, availability, and are supported by free development systems from their manufacturers.

2.1 Lectures and laboratory exercises

The subject has two hours of lectures and two hours of laboratory exercises per week in a 13-week semester.

2.1.1 Curriculum of lectures (13 weeks)

1. Introduction, history and generations of PLD devices, usage of PLD devices
2. Architectures of SPLD, CPLD and FPGA. Basics of VHDL language, syntax, entity, architecture
3. Concurrent statements – unconditional, conditional and selective statements, components, processes
4. MUX (with when-else, with-select, process-case), process, sensitivity list, wait statement
5. Implementation of RS, D, T, JK flip-flops, parameterization by generate and loop statement
6. Architecture and features of modern CPLD devices
7. Implementation of memories (sync/async. ROM, single and dual-port RAM, FIFO)
8. Architectures and features of modern FPGA devices
9. Automated test benches, simple UART (8D, 1S, 0P)
10. Attributes of types, subtypes, arrays, signals, entities, user defined attributes
11. Libraries, packages, Library of Parameterized Modules (LPM 2 0 0)
12. Description of Moore and Mealy state machines in VHDL, file types
13. Implementation and usage of software processors in FPGA devices

2.1.2 Laboratory exercises

Professional tools for digital design in the VHDL language are expensive, but free versions which have some functionality limitations can be obtained from web sites of most CAE/CAD/EDA companies and PLD manufacturers for free. Thanks to this we are using for practical lessons free version of the VHDL simulator "VHDL Simili" from Symphony EDA. The free version

has significant advantages and disadvantages. The advantage of the VHDL Simili is a very intuitive program to work with and availability of a free version, so the students get familiar with it in a short time, and they can use it for learning VHDL language and working on semester projects at home. The disadvantage of the free version of VHDL Simili is that it cannot plot VHDL variables and the maximum number of waveforms is limited. Smaller disadvantage is that it has reduced simulation speed for larger designs.

In practical lessons the students work on a given task, describe it in VHDL, simulate it on VHDL simulator, and in several tasks they verify the design on real hardware. For testing of the designs on real hardware we use the Design Laboratory Package UP1 boards from Altera. We obtained these boards at a very low price as a benefit of the Altera University Program. This board meets all needs for teaching the digital logic design with the state-of-the-art development tools and programmable logic devices. It contains the FPGA device FLEX10K20 and CPLD device MAX7128S from Altera, an oscillator, DIP switches, push buttons, LEDs, seven-segment LED displays, a VGA connector, a PS/2 connector, a configuration device and expansion slots. For the implementation of digital designs we use the Altera Quartus II Web Edition software, also free of charge.

For the software verification of the laboratory designs use the VHDL test benches written by the students or prepared by the lecturer. Possibility of simulation files generated by waveform editors is only mentioned but not used because students get better knowledge of the VHDL language when they use it for all purposes – design and verification. Test benches written in VHDL have several advantages compared to waveform based simulation:

- VHDL test bench is more concise
- VHDL test bench can provide an automatic test of the correctness of simulation results
- VHDL test bench can read and write input/output data to or from files for the design under test
- Test bench written in VHDL is independent of the type of currently used development / simulation system
- Test bench written in a high level language is a suitable method for testing of large designs

There are three alternatives of the VHDL digital designs testing:

- Functional simulation based on .vhd files
- Post-fit functional simulation based on .vhd and .vho files
- Timing simulation based on .vhd, .vho and .sdo files
- The first alternative is used in most cases, the second and third is used before the verification of the digital designs on real devices. All these three possibilities of testing are supported by the VHDL Simili simulator.

2.1.3 Example of testbench

As an example of a testbench used in the exercises, the testbench for the Fibonacci numbers generator follows:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity tb_fib_gen is
end tb_fib_gen;

architecture behav of tb_fib_gen is
   component fib_gen
      port (
         clk : in std_logic;
         reset : in std_logic;
         result : out std_logic_vector(7 downto 0);
         overflow : out std_logic
      );
   end component;
   signal clk, reset, overflow : std_logic;
   signal result : std_logic_vector(7 downto 0);
begin
   dut : fib_gen
      port map (clk, reset, result, overflow);
   testbench : process
      procedure do_clk is
      begin
         clk <= '0';
         wait for 20 ns;
         clk <= '1';
         wait for 20 ns;
      end;
      type tf_text is file of string;
      file fw : tf_text;
      variable tb_current, tb_last : integer;
      variable tb_result, int_res : integer;
   begin
      file_open(fw, "result.txt", write_mode);
      -- synchronous reset
      reset <= '1';
      do_clk;
      reset <= '0';
      -- test of first element of sequence
      int_res := to_integer(unsigned(result));
      write(fw, integer'image(int_res) & " ");
      if (int_res /= 0) then
         assert false report
            "Error at #1"
            severity error;
      end if;
      do_clk;
      -- test of second element of sequence
      int_res := to_integer(unsigned(result));
      write(fw, integer'image(int_res) & " ");
      if (int_res /= 1) then
         assert false report
            "Error at #2"
            severity error;
      end if;
      -- test of others elements of sequence
      tb_current := 1;
      tb_last := 0;
      for i in 3 to 15 loop
         do_clk;
         int_res := to_integer(unsigned(result));
         write(fw, integer'image(int_res) & " ");
         tb_result := tb_current + tb_last;
         tb_last := tb_current;
         tb_current := tb_result;
         if (overflow = '0') then
            if (int_res /= tb_result) then
               assert false report
                  "Error at #" & integer'image(i)
                  severity error;
            end if;
```

```
         end if;
      end loop;
      file_close(fw);
      assert false report
         "End of testbench."
         severity note;
      wait;
   end process;
end behav;
```

2.1.4 Example of digital design

The Fibonacci numbers generator used as the design under test is described by the following listing:

```
-- fib_gen.vhd
-- Generator of Fibonacci numbers
-- 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity fib_gen is
   port (
      clk : in std_logic;
      reset : in std_logic;
      result : out std_logic_vector(7 downto 0);
      overflow : out std_logic
   );
end fib_gen;

architecture struct of fib_gen is
   signal last_val : unsigned(8 downto 0);
   signal curr_val : unsigned(8 downto 0);
   signal sum : unsigned(8 downto 0);
begin
   -- register for current value
   process (clk, reset)
   begin
      if (clk'event and clk = '1') then
         if (reset = '1') then
            curr_val <= to_unsigned(1, 9);
         else
            curr_val <= sum;
         end if;
      end if;
   end process;

   -- register for last value
   process (clk, reset)
   begin
      if (clk'event and clk = '1') then
         if (reset = '1') then
            last_val <= (others => '0');
         else
            last_val <= curr_val;
         end if;
      end if;
   end process;

   -- adder
   sum <= last_val + curr_val;

   result <= std_logic_vector(last_val(7 downto 0));
   overflow <= last_val(8);
end struct;
```

2.1.5 Compilation and timing simulation

After successful functional simulation by the testbench described in section 2.1.3, the design is compiled by the Quartus II software for the Altera EPM7128SLC84-7 device. The development system Quartus generates a standard netlist VHDL Output File (.vho) and a Standard Delay Format Output File (.sdo). A timing simulation, using again the test bench in conjunction with previously generated .vho and .sdo files, can then be performed. The result of the timing simulation in VHDL Simili is shown in Fig. 1.

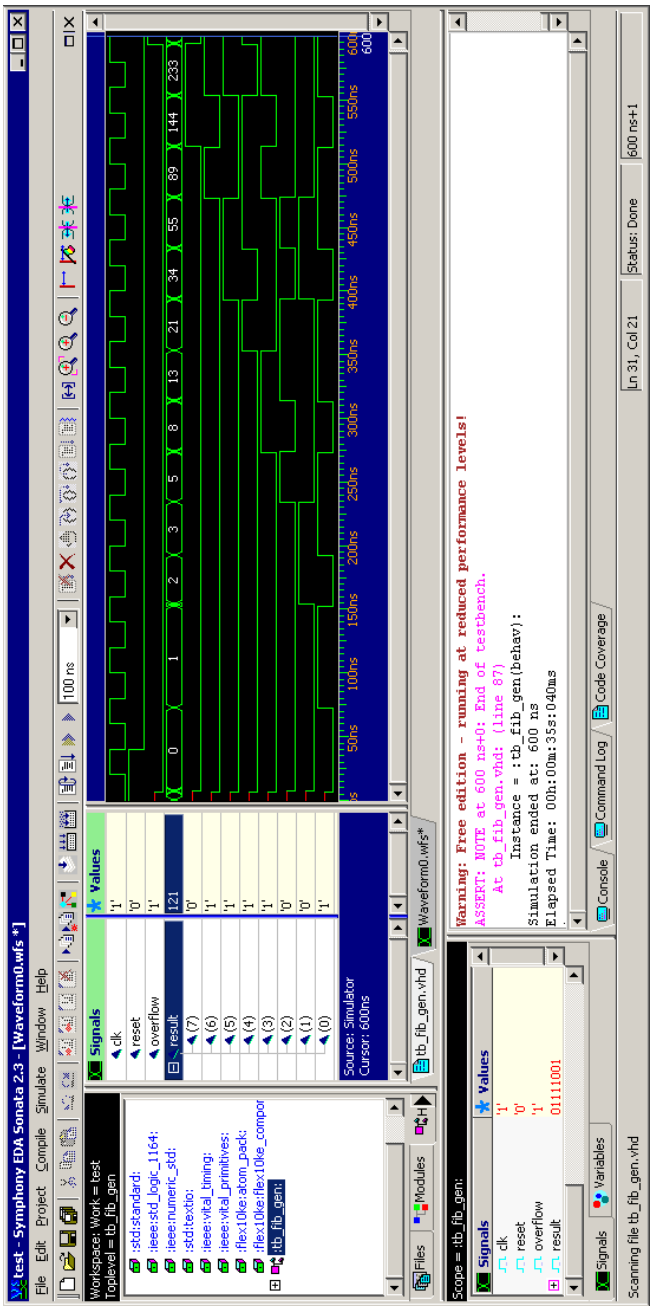

Fig. 1: Screen shoot of timing simulation

3 Conclusion

Subject Programmable Logic Devices is now successfully established in study programmes of Faculty of Electrical Engineering and gives clear and logical reading of basic terms of PLDs, digital designs and VHDL language to the students.

References:

[1] Poupa, M.: *Set of tasks for practice lessons of VHDL language*, University of West Bohemia, 2004

[2] Poupa M., Holota R.: Teaching of programmable logic devices, *Proceedings of the "Applied Electronics 2001" international conference*, 2001, pp. 91-92