Using HDL Translators in VLSI Design Laboratory Exercises

Atanas Nikolov Kostadinov

Abstract – **There exist many different hardware description languages (HDLs). The most popular of them are VHDL, Verilog and SystemC. The most engineers have more experience in one of them. The using of a HDL translator can be presented in VLSI design laboratory exercises. At least one lecture about HDL translators and about explanation of some keywords in another (received after translation) hardware description language is a topic suitable to be included in VLSI design discipline.**

Keywords – **HDL translators, VLSI design, Laboratory exercises**

I. Introduction

There exist many different HDLs (Hardware description languages) as VHDL (Very High Speed Integrated Circuit Description Language), Verilog, SystemC, ABEL (Advanced Boolean Expression Language), AHDL (Altera HDL), Handel-C, Lola, etc [1]. The most popular of them are VHDL, Verilog and SystemC.

Usually, in the VLSI (Very-large-scale integration) design subject one HDL has been presented by the lecturer. In Technical University – Sofia, Plovdiv branch VHDL is used. In he VLSI design an important issue is to be used various IPs (Intellectual Properties) modules [2, 3, 4]. If there exist HDL IP in a language which designer is not proficient enough then the HDL translator can be used. Other advantages of using above-mentioned translators are given on the next lines [5, 6].

- Double your market share by supporting both VHDL and Verilog languages. Instead of investing weeks of non-recurring engineering time to develop designs or models in the second language, use v2v translators to do it for you in a fraction of the time;
- Save time in maintenance by keeping designs in one language and translating changes into the other;
- Learn language translation shortcuts and tips as information warnings guide the user. Sometimes a very minor change to the source will make the translation go much smoother. V2V translators exploit commonality between the two languages;
- Decrease language learning time as translators acquaint designers with both Verilog and VHDL. Most engineers have more experience in one of the languages. Using the translation tools builds on what you already know and teaches you as you need to learn. Any new language appears difficult at the start. The translation tools make learning much easier;
- Make conversions easy and consistent. Single point of control is the key to successful change procedures;
- Converting several thousand lines of VHDL code into Verilog by hand can take hundreds of engineering man-hours. HDL translator can do the job in a matter of minutes significantly reducing workload, engineering costs, and time. The cost of the program can be recovered after its single run;
- In a complex code comments are crucial to the continuing maintenance of the design. HDL translators usually preserve comments and formatting of the original source code;
- A translator that could translate all possible language constructs would be cumbersome and costly. The part of source code it cannot translate (usually very small or none) will be commented out, for subsequent manual translation;
- The HDL translators are simple to use;

A. Kostadinov is with the Department of Computer Systems and Technologies, Technical University - Sofia, Plovdiv branch, 25 Tsanko Diustabanov Str., 4000 Plovdiv, Bulgaria, e-mail: kostadat@tu-plovdiv.bg

II. Using HDL Translators in VLSI Design Laboratory Exercises

There are realized many different HDL translators [5, 6, 7, 8, 9]. In this paper X-HDL is used in demo mode and with an evaluation license. The demo mode is fully functional but restricts the source file size to 1kB [9] which is suitable for educational purposes.

In VLSI design laboratory exercises X-HDL can be used to translate the designed by students combinational and sequential hardware components according VLSI design laboratory exercises guide [10].

A. Using X-HDL in translation of combinational hardware components

As an example, the VHDL code which will be translated is describing a multiplexer (MUX). This MUX consists of four 3-bit inputs called I0, I1, I2, and I3, 2-bit selection signal S and 3-bit output signal named O. The code before translation is shown on the next lines.

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

entity Mux is
port(	I3:	in std_logic_vector (2 downto 0);
I2:	in std_logic_vector (2 downto 0);
I1:	in std_logic_vector (2 downto 0);
I0:	in std_logic_vector (2 downto 0);
S:	 in std_logic_vector (1 downto 0);
O:	out std_logic_vector (2 downto 0));
end Mux;

architecture beh of Mux is
begin
process(I3,I2,I1,I0,S)
begin
```

```
case S is
when "00" =>      O     <= I0;
when "01" =>      O     <= I1;
when "10" =>      O      <= I2;
when "11" =>      O       <= I3;
when others =>    O      <= "ZZZ";

end case;
end process;
end beh;
```

The received Verilog code after translation with a small manual correction is shown on the lines bellow.

```
`timescale 1 ns / 1 ns

module Mux (I3, I2, I1, I0, S, O);
input [2:0]  I3;
input [2:0]  I2;
input [2:0]  I1;
input [2:0]  I0;
input [1:0]  S;
output [2:0] O;
reg [2:0]    O;

always @ (I3 or I2 or I1 or I0 or S)
case (S)
2'b00 :   O  <= I0;
2'b01 :   O  <= I1;
2'b10 :   O  <= I2;
2'b11 :   O  <= I3;
default : O  <= 3'bZZZ;

endcase
endmodule
```

The received Verilog code is simulated in order to be checked the correct work of this module. It is used ModelSim Altera Starter Edition [11]. The obtained results showing the correct work of the translated module are presented in Fig. 1.

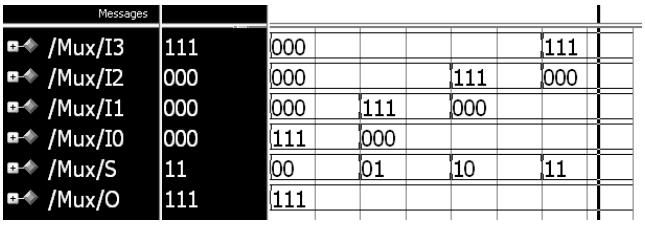

FIG. 1. MUX SIMULATION RESULTS

B. Using X-HDL in translation of sequential hardware components

In the next example is used VHDL code of a counter. This 8-bit counter has clock input CLK, reset signal RST and 8-bit output signal named Q. The initial VHDL code is presented on the next lines.

```
library   ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity counter is
port (CLK: in std_logic ;
RST: in std_logic;
Q: out std_logic_vector (7 downto 0));
end counter;

architecture beh of counter is
signal countL : std_logic_vector (7 downto 0);
begin
process (CLK, RST)
begin
if RST ='1'
then countL <= "00000000";
elsif (CLK'event and CLK ='1')
then  countL <= countL + 1;

end if;
end process ;
Q  <=  countL;
end beh ;
```

The translated Verilog code with a small correction manual correction is given on the lines bellow.

```
`timescale 1 ns / 1 ns

module counter (CLK, RST, Q);
input       CLK;
input       RST;
output [7:0] Q;
reg [7:0]    countL;

always @ (posedge CLK)
if (RST == 1'b1)
countL <= 8'b00000000;
else countL <= countL + 1;
assign Q = countL;

endmodule
```

A part of simulation results of the obtained Verilog description of the counter is presented in Fig. 2.

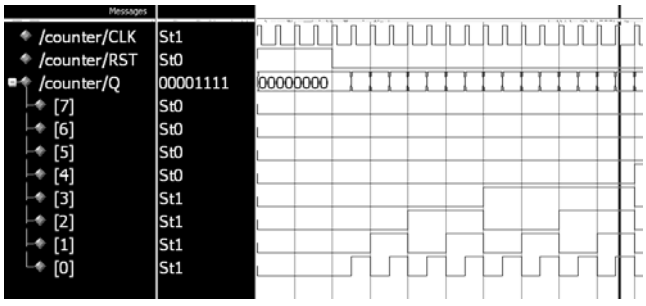

FIG.2. COUNTER SIMULATION RESULTS

C. Using X-HDL in translation of structural described hardware

In VLSI design, the structural descriptions are applied when we have circuits based on different hardware components. A simple example is presented in Fig. 3. This circuit has been realized using EAGLE Layout Editor [12].

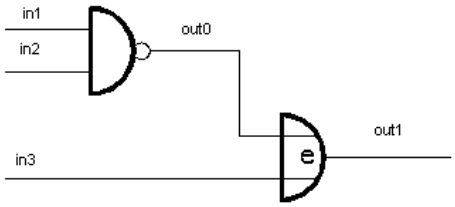

FIG.3. EXAMPLE CIRCUIT

In this example the circuit consists of two components: a 2-input NAND gate and 2-input XOR gate. The truth table is introduced in Table 1.

TABLE 1. TRUTH TABLE

in3	in2	in0	out1
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

The circuit is described by VHDL using structural type of description. The X-HDL translator is used and Verilog description is received. After a small manual correction the Verilog source files are simulated. The received results after simulation are presented in Fig. 4.

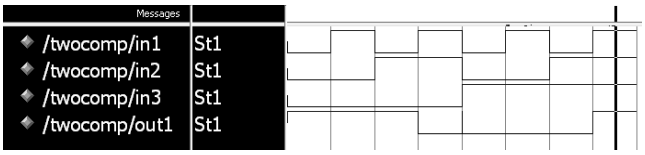

FIG. 4. RESULTS AFTER SIMULATION

We can see that results from truth table and received after simulation are exactly the same. In this way, we can conclude that the translation from VHDL to Verilog is done correctly.

III. USING HDL TRANSLATORS IN VLSI DESIGN RESEARCH AND DEVELOPMENT

Usually, for the research and development is used bigger than 1kB HDL design files. To translate them from VHDL to Verilog using X-HDL software tool is required a license. It is possible to be requested a 10-day evaluation license to X-Tek Corporation in order to cope with the larger source files.

During the period of my ERCIM (European Research Consortium for Informatics and Mathematics) fellowship a predicate logic processor has been synthesized [13]. This is an ASIP (Application-Specific Instruction-Set Processor) type of processor. The simplified block diagram of the designed architecture is shown in Fig. 5.

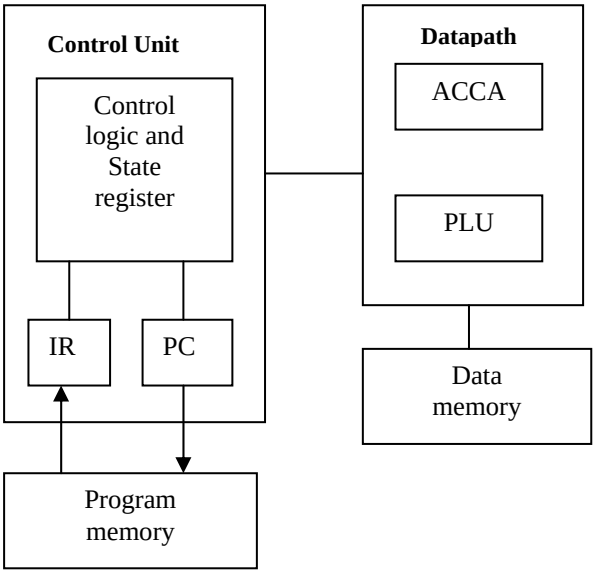

FIG. 5. PLP SIMPLIFIED BLOCK DIAGRAM

In Fig. 5, the abbreviation IR stands for Instruction Register, PC – Program Counter, ACCA – Accumulator named A, PLU – Predicate Logical Unit.

Then the modification called PBOP processor has been proposed and designed [14]. This processor architecture is oriented to work with binary numbers in predicate logic. It has been implemented the search and sorting algorithms. The PBOP VHDL design files have been translated in Verilog. Then Quartus ® II project has been realized. After successful compilation, the FPGA (Field Programmable Gate Array) part of the DE2 (Development and Education 2) board [11] has been configured.

As the latest design step, the PBOP processor has been verified using SignalTap® II embedded logic analyzer which is included in Quartus ® II software tool. A part of the registered signals is presented in Fig. 6. The proposed processor is clocked with a frequency of 50 MHz (PIN_N2 of the DE2 board is connected to cpu_clk) during the verification process. The data are captured using the clock signal as an acquisition signal and the reset signal as a trigger signal. The obtained information is the same as information when is used VHDL design files. This shows again that the translation from VHDL to Verilog is correct.

The signals presented in Fig. 6 are address (RAM address bus), data_in and data_out (RAM data bus), Mre and Mwe (Signals required for the realizing of the reading and writing to the RAM), OPCODE (Code of the proposed and implemented instructions), ICout (Content of Index Counter), EQUAL, GRATER, LESS (Used flags), PCout (Content of Program Counter), and Q (a binary value loaded to accumulator named A).

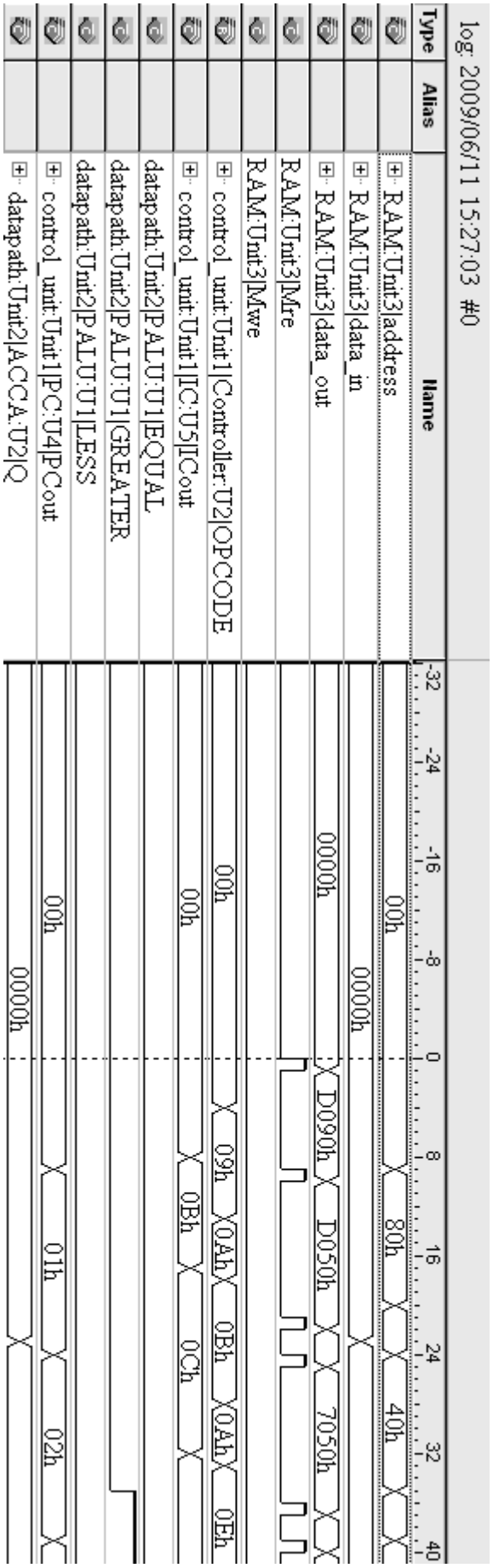

FIG. 6. SIGNALTAP II RESULTS

IV. CONCLUSIONS

The HDL translators are very useful in VLSI design both in the higher education as well as in the research and development.

The software tool translated from VHDL to Verilog and back especially the used in this paper X-HDL has done the translation in a correct way. There are parts of the produced code where are required small manual corrections.

This VLSI design laboratory exercise should be preceded by a lecture about HDL translators and about explanation of some keywords in another (received after translation) hardware description language.

REFERENCES

[1] http://en.wikipedia.org/wiki/Hardware_description_language
[2] A. Kostadinov. *Embedded System Design Based on IP (Intellectual Property) Blocks*, Proceedings of The National Conference ELECTRONIKA'2006, pp. 150-155, 2006.
[3] J. Pimentel, L. Hoang. *A VHDL Library of IP Cores for Power Drive and Motion Control Applications*, Proceedings of Canadian Conference on Electrical and Computer Engineering, Vol. 1, pp. 184-188, 2000.
[4] http://www.opencores.org/
[5] http://www.veritools-usa.com/vhdl2verilog.shtml
[6] http://www.trilent.net/products/vhdl2v/index.html
[7] http://www.ocean-logic.com/
[8] http://www.syncad.com/verilog_vhdl_translator.htm
[9] http://www.x-tekcorp.com/xhdl.php
[10] A. Kostadinov, D. Manova. *VLSI design laboratory exercises guide*, TU-Sofia, Plovdiv branch, Plovdiv, 2005 (in Bulgarian).
[11] http://www.altera.com/
[12] http://www.cadsoftusa.com/
[13] A. Kostadinov, G. Kouzaev. *Predicate Logic Processor of Spatially Patterned Signals.* Proceedings of The First WSEAS Int. Conference on Multivariate Analysis and its Application in Science and Engineering, pp. 94-96, 2008.
[14] A.Kostadinov, G. Kouzaev. *Predicate and Binary Operations Processor*, Proceedings of The Eighth WSEAS Int. Conference on Application of Electrical Engineering, pp. 199-204, 2009.