Printed Circuit Board Design with HDL Designer

Tom Winkert
NASA/Goddard Space Flight Center
301-286-2917
tom.winkert@nasa.gov

Teresa LaFourcade
NASA/Goddard Space Flight Center
301-286-0019
teresa.l.lafourcade@nasa.gov

1 Abstract

Staying up to date with the latest CAD tools both from a cost and time perspective is difficult. Within a given organization there may be experts in Printed Circuit Board Design tools and experts in FPGA/VHDL tools. Wouldn't it be great to have someone familiar with HDL Designer be able to design PCB's without having to learn another tool? This paper describes a limited experiment to do this.

2 PCB Design Process

The PCB design process consists of three phases. Schematic capture, input of component information, and netlist conversion.

3 Schematic Capture

Most designers typically use Mentor Board Station or DxDesigner. These tools have robust front-end schematic capture features. For small applications there may be another alternative. For this project HDL Designer version 2004.1 was used. The HDL Designer block diagram editor was used for schematic capture. The HDL Designer symbol editor created the individual component symbols. This feature allows someone already familiar with the tool to design a PCB. The HDL designer tool can be used in conjunction with the Modelsim simulator for PCB simulation. An FPGA designer can now work under the same simulation environment for the PCB that is currently utilized for the FPGA.

Two part symbols are shown in Figures 1 and 2. Each of these symbols displays the part number, the package type, the layout of the pin numbers, and the signal list. VHDL generics, explained in the next section, were used to hold and display this component information.

Figure 3 is a block diagram or schematic containing two part symbols connected together as well as to VCC and GND. The parts are an AC08 (shown in Figure 2) and an AC04. The reference designators I0 and I1, which are displayed on the schematic, are automatically assigned to the parts by the program.

.

4 VHDL

The plan is to use VHDL for the PCB netlist format. Very little information explaining how to accomplish this is available. A major effort was taken to standardize VHDL descriptions of not only physical component package information, such as pin and part numbers, but also digital timing, AC, and DC characteristics as well. This standard was eventually abandoned.

Various approaches were studied on how to convey this information into the VHDL net list primarily focusing on pin numbers since that would be essential to the netlist creation. Possible approaches included:

- VHDL comments
- VHDL attributes
- VHDL generics

Although VHDL comments would work, it is more desirable to have this information as part of the language since there may be future extensions to this experiment that would need to use the part information for simulation. The simulator does not read comments.

VHDL user defined attributes were logically a good choice since part information can be used in the component code and passed on to the board schematic. This approach turned out to be unfeasible because HDL designer does not display component attributes on its block diagrams. If these block diagrams are to be used as schematics they must display component pin numbers.

VHDL generics can be displayed on the block diagrams so that approach was chosen. Generics will display the pin numbers as a large block of text rather than on individual ports (see Figure 2).

5 Net list Conversion

PCB layout tools do not usually import VHDL net lists. Even if such a tool did exist there is no standard VHDL netlist format. A typical PADS format was chosen for PCB layout. The challenge is now to take the VHDL net list and convert it to the PADS format. A search for conversion utilities turned up one company that specialized in doing just that. EDAX-NET from Conversion Factors [1] has converted many different formats including VHDL. Their VHDL conversion only assigned part package pin numbers based upon the ordering in the VHDL component declaration. Some of the drawbacks to this approach are:

-Every pin on a part in the component declaration (such as no connects) may not need to be listed.

-Ports need to be listed in the same order in the component declaration and on the part.

-Multiple pin numbers cannot be used for a single port (such as power and ground).

Conversion Factors then modified their utility to eliminate those drawbacks. Their utility now inputs generic mapping information for part pin numbers, part numbers and part package type. This information combined with the port mapping enables EDAX-NET to produce a PADS compatible net list. The test circuit VHDL shown in Figure 3 is listed in Appendix A. The output PADS net list is Appendix B.

5 Conclusion

This study has shown schematic capture using HDL designer and net list conversion to create a PCB netlist. Two test boards are currently in the design phase that is using HDL Designer to create actual hardware. The approach described here is not envisioned as a replacement for current PCB design tools. Obviously, those tools have many features that would take a substantial overhaul of HDL designer to begin to compare. However, some improvements could be made to make this an attractive alternative to many customers. Improvements could be in schematic display, adding PCB net list output format options, and some level of design rule checking, and part list generation.

6 References

[1] Conversion Factors, Inc. 918-825-9300
http://www.confac.com

7 Acronyms

CAD	Computer Aided Design
EIA	Electronic Industries Alliance
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
PCB	Printed Circuit Board
VHDL	VHSIC Hardware Description Language
GND	ground
VCC	power
PADS	Mentor Trade Mark

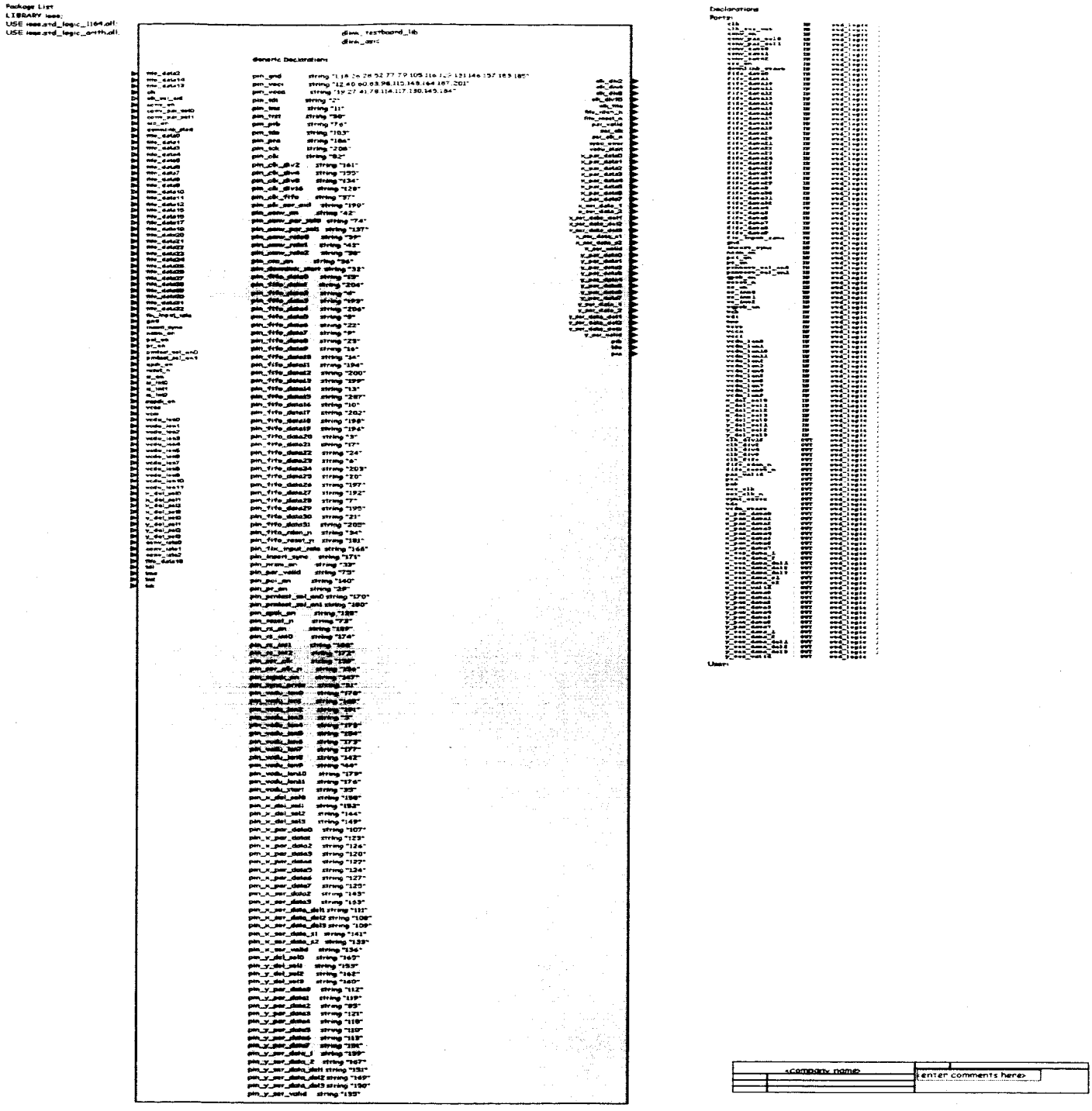


Figure 1.

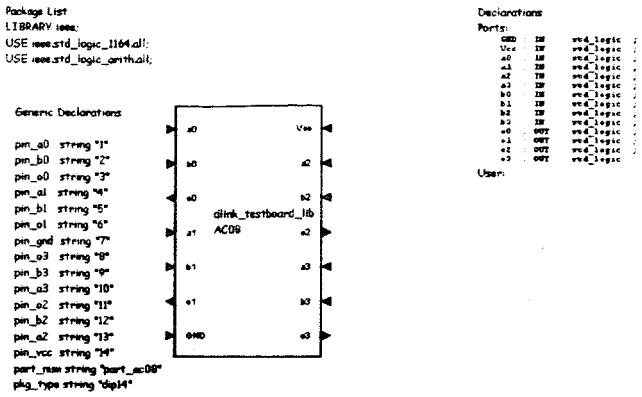


Figure 2.

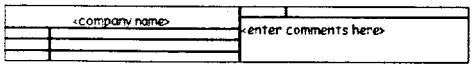
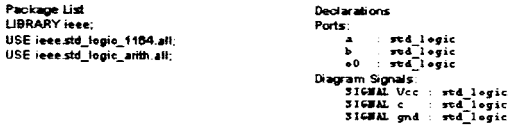
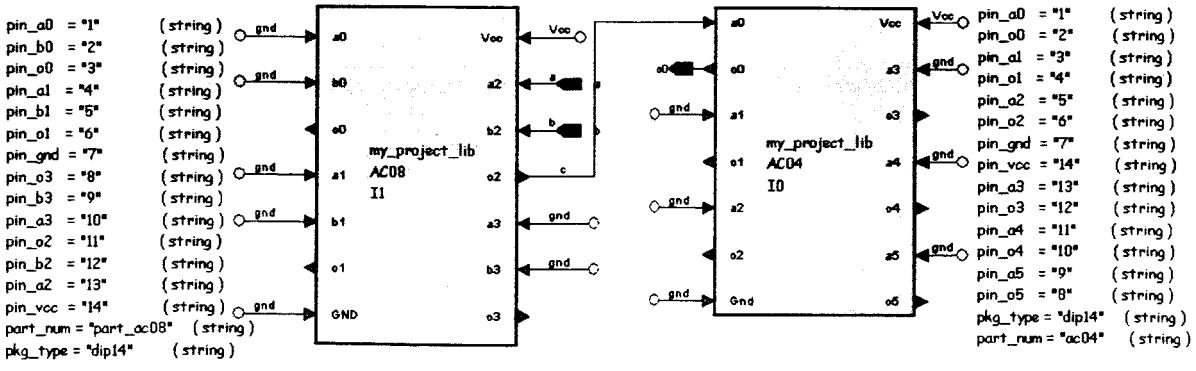
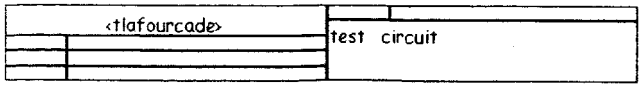


Figure 3.







Appendix A

```
-- VHDL Entity my_project_lib.test_ckt.symbol
--
-- Created:
--        by - tlafourca (TERESANT1)

--
-- Generated by Mentor Graphics' HDL Designer(TM) 2004.1
--
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY test_ckt IS
  PORT(
    a  : IN    std_logic;
    b  : IN    std_logic;
    o0 : OUT   std_logic
  );

-- Declarations

END test_ckt ;

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;


ARCHITECTURE struct OF test_ckt IS

  -- Internal signal declarations
  SIGNAL Vcc : std_logic;
  SIGNAL c   : std_logic;
  SIGNAL gnd : std_logic;


  -- Component Declarations
  COMPONENT AC04
  GENERIC (
    pin_a0   : string := "1";
    pin_o0   : string := "2";
    pin_a1   : string := "3";
    pin_o1   : string := "4";
    pin_a2   : string := "5";
    pin_o2   : string := "6";
    pin_gnd  : string := "7";
    pin_vcc  : string := "14";
    pin_a3   : string := "13";
    pin_o3   : string := "12";
    pin_a4   : string := "11";
    pin_o4   : string := "10";
    pin_a5   : string := "9";
    pin_o5   : string := "8";
    pkg_type : string := "dip14";
    part_num : string := "part_ac04"
  );
  PORT (
    Gnd : IN    std_logic ;
    Vcc : IN    std_logic ;
    a0  : IN    std_logic ;
    a1  : IN    std_logic ;
    a2  : IN    std_logic ;
```

```
    a3 : IN    std_logic ;
    a4 : IN    std_logic ;
    a5 : IN    std_logic ;
    o0 : OUT   std_logic ;
    o1 : OUT   std_logic ;
    o2 : OUT   std_logic ;
    o3 : OUT   std_logic ;
    o4 : OUT   std_logic ;
    o5 : OUT   std_logic
  );
  END COMPONENT;
  COMPONENT AC08
  GENERIC (
    pin_a0  : string := "1";
    pin_b0  : string := "2";
    pin_o0  : string := "3";
    pin_a1  : string := "4";
    pin_b1  : string := "5";
    pin_o1  : string := "6";
    pin_gnd : string := "7";
    pin_o3  : string := "8";
    pin_b3  : string := "9";
    pin_a3  : string := "10";
    pin_o2  : string := "11";
    pin_b2  : string := "12";
    pin_a2  : string := "13";
    pin_vcc : string := "14";
    part_num : string := "part_ac08";
    pkg_type : string := "dip14"
  );
  PORT (
    GND : IN    std_logic ;
    Vcc : IN    std_logic ;
    a0 : IN    std_logic ;
    a1 : IN    std_logic ;
    a2 : IN    std_logic ;
    a3 : IN    std_logic ;
    b0 : IN    std_logic ;
    b1 : IN    std_logic ;
    b2 : IN    std_logic ;
    b3 : IN    std_logic ;
    o0 : OUT   std_logic ;
    o1 : OUT   std_logic ;
    o2 : OUT   std_logic ;
    o3 : OUT   std_logic
  );
  END COMPONENT;

BEGIN

  -- Instance port mappings.
  I0 : AC04

    PORT MAP (
      Gnd => gnd,
      Vcc => Vcc,
      a0  => c,
      a1  => gnd,
      a2  => gnd,
      a3  => gnd,
      a4  => gnd,
      a5  => gnd,
```

```
      o0  => o0,
      o1  => OPEN,
      o2  => OPEN,
      o3  => OPEN,
      o4  => OPEN,
      o5  => OPEN
    );
  I1 : AC08
    PORT MAP (
      GND => gnd,
      Vcc => Vcc,
      a0  => gnd,
      a1  => gnd,
      a2  => a,
      a3  => gnd,
      b0  => gnd,
      b1  => gnd,
      b2  => b,
      b3  => gnd,
      o0  => OPEN,
      o1  => OPEN,
      o2  => c,
      o3  => OPEN
    );
END struct
```

Appendix B

PADS-LOGIC*
REMARK created with EDAX-NET V7.0.3
REMARK

PART
I0 PART_AC04,AC04@DIP14
I1 PART_AC08,AC08@DIP14

NET
SIGNAL GND
I0.7 I0.3 I0.5 I0.13 I0.11 I0.9 I1.7 I1.1
I1.4 I1.10 I1.2 I1.5 I1.9
SIGNAL VCC
I0.14 I1.14
SIGNAL C
I0.1 I1.11
SIGNAL O0
I0.2
SIGNAL A
I1.13
SIGNAL B
I1.12

END OF ASCII OUTPUT FILE