APPLICATION NOTE

AN059
Mentor Graphics Design Flow for targeting Philips CPLDs

1996 Sep 27

Philips
Semiconductors

PHILIPS

Mentor Graphics Design Flow for targeting Philips CPLDs

AN059

INTRODUCTION

The Programmable Logic Group of Philips Semiconductor is developing a family of advanced 3-volt and 5-volt complex programmable logic devices(CPLDs). The XPLA series, designated as the PZ5000 - (5-volt) and PZ3000 (3-volt) series devices, is footprint compatible with the Altera 7000 series devices. The principle advantage of Philips CPLDs over all existing CPLDs is that they consume zero static power. The other advantages are 25% higher logic capacity and a better ability to fit logic with fixed pinouts. The first devices, the 32-macrocell PZ3032 and PZ5032, began sampling in Feb 1996. The PZ3064/PZ5064 iand the PZ5128/PZ3128 are scheduled to sample in Q4 1996. The PZ5128/PZ3128 are in-system programmable. All devices are all programmable on Data I/O and BP Microsystems programmers.

Minc Inc has developed fitters for the PZ5000/PZ3000 series for up to 128 macrocells. This allows workstation users to target Philips CPLDs within workstation environments. The software is capable of automatically partitioning across multiple CPLDs. Verilog and VHDL models are generated for timing simulation and post fit board-level simulation.

This note provides scripts for using this capability. The Minc fitter can be used with most workstation flows which use VHDL or Verilog from Cadence, Synopsys, Mentor Graphics, and Exemplar Logic. It can be used with Composer and Concept schematic editors from Cadence and Design Architect from Mentor. In this application note, an example of a design flow for using Mentor Graphics softwarefor simulation and compilation of VHDL designs is given. This flow can be used with minor edits for Verilog synthesis.

For additional information, telephone Philips Applications Support at 888-coolpld or browse http://www.coolpld.com.The following documentation is available either through the web server or telephoning 888 coolpld.

PLDesigner-XL User’s Guide

PLDSynthesis II User’s Manual

PZ5000/PZ3000 Series Data Sheets

DESIGN FLOWS

The software needed to target Philips CPLDs is available from Mentor Graphics. The software required depends on the design flow. This software should be installed as provided in Chapter 1 in the PLDSII User’s Manual.

In the steps listed below, $1 is used to represent the design name, and $_tbthe testbench name. Generally, there are a number of different methods to design using Mentor tools , and scripts will usually vary based on the design.

Mentor Graphics may have access to different tools, including

Synthesis

Autologic - VHDL or Verilog

Synopsys - VHDL or Verilog

Exemplar - VHDL or Verilog

Simulation

Quicksim

QuickVHDL

This note provides a design flow for using the Autologic synthesis tool. The flow and examples for both Quicksim and QuickVHDL simulators are given.

Synthesis using the Autologic Flow

The steps given below do the following:

1. Setup environment
2. Create library and compile VHDL source.
3. Synthesize using Autologic.
4. Open and check the schematic in Design Architect.
5. Generate and check the symbol.
6. Export to PDSII to generate a $1.src file
7. Compile the edif file to a jedec file and produce a delay-annotated VHDL model from the jedec file.
8. Simulate using Quicksim or Quickvhdl.

To use Autologic for Philips CPLDs, the mgc_location_map in the user’s home or project directory should contain the following :

```
MGC_LOCATION_MAP_2
#MGC Synlib
$MGC_SYNLIB
/export/home2/lib/synlib
/tmp_mnt/export/home2/lib/synlib
$MGC_GENLIB
/export/home2/lib/gen_lib
/tmp_mnt/export/home2/lib/gen_lib
$MINC_PATH
#$PLDS2_SYN_LIB
```

The flow for Mentor tools is as follows, broken down into Quickvhdl, Autologic, Design Architect, PLDSII, and Quicksim functions.:

Quickvhdl steps

```
mkdir src
cp $1.vhd src
export MGC_LOCATION_MAP=/export/home/lss/designs/mgc/mgc_location_map
qvlib work
qvmap work“<project_directory>/work”
qvcom src/$1.vhd -work work -synth
```

In some cases, testbenches do not compile with the -synth option to the qvcom command.

The flow for a functional simulation in Quickvhdl is as follows. Although recommended, a test bench is not required.

```
qvcom $1_tb.vhd -work designs
qvsim -lib <library> <entity>
File-Load testbench
view wave signals list
```

Autologic steps

To invoke Autologic to use the command line intrface, enter

```
alui -nodisplay
```

The autologic session will include the following steps.

```
opn design -vhdl <path>/src/$1.vhd
env dst gen_lib
syn vhdl <library> <entity> -arch <architecture>
opt area -low
sav design -eddm -model <entity> -map <library> eddm -schematic
quit -f
```

Design Architect steps

To invoke Design Architect, enter

```
da &
```

The steps in Design Architect are :

open sheet in eddm

check schematic

generate and check symbol

re-open schematic

export to pldsII produces *.src in eddm/design/minc directory

PLDSII steps

Either the PLDesigner graphical user interface or a script can be used. For use ot the PLDesigner GUI, see the PLDesigner User Manual. As of this writing, Philips CPLDs are not released with PLDSII. Below is a script for compiling to Philips CPLDs which requires release of only Philips device files. These are available through Mentor Graphics. To run the script, enter

```
minc.script $1
```

The contents of minc.script are

```
cp $MINC_PATH/default.pi .
cp $MINC_PATH/default.cst .
mv default.pi $1.pi
mv default.cst $1.cst
make_src $1.edf
plcomp $1.src
#plsim $1.stm
plopt $1.afb
plscan $1
plfit $1
plfuse $1
pldoc $1
modgen $1.j1 pz3032-8-qfp44 -vhdl
```

This produces a jedec file ($1.j1) and a delay annotated vhdl model ($1.vho). The jedec file can be used to program a PZ3000 or PZ5000 series device[1]. This file can be read into Quickvhdl but there may be modifications to the toriginal testbench for a simulation.

Selecting a Philips CPLD

The Philips CPLD used is specified in the <design>.pi and <design>.cst files. This allows a user to direct PLDesigner to either target a specific device as the PZ3032 or to scan all devices and provide multiple solutions. The use of these files is described in detail in Chapters 14-16 of the PLDesigner-XL User's Guide. To target the PZ3032, the following can be used .

<design>.cst

```
TEMPLATE =  XPLA32_32 or XPLA64_64 ;
```

<design>.pi

```
DEVICE
TARGET 'TEMPLATE XPLA32_32 TQFP-44-P32';
default ;
END DEVICE;
```

Quicksim II simulation steps

From the <design>/eddm/<design> directory, enter quicksim -timing_mode typ plds2_vpt

File-Open design_sheet

Select signals to be monitored in schematic

Invoke Trace, List

[1] Depending on the .pi file, PLDesigner can partition a design across multiple devices, so that $1.j1, $1,j2,... are produced.

Generate Stimuli

Run

DESIGN FLOW EXAMPLE

An example of a flow using Autologic, Design Architect, and either Quicksim II or the Quickvhdl simulator is given below. The example is an 4 to 1 multiplexer of 6-bit busses.

The vhdl source is

```
-- Philips CPLD Applications
-- 6-bit 4 to 1 multiplexer
-- August 20, 1995
library ieee ;
use ieee.std_logic_1164.all ;
entity m41 is
port (a,b,c,d: in std_logic_vector (5 downto 0);
 sel: in std_logic_vector (1 downto 0);
 z : out std_logic_vector (5 downto 0));
end m41 ;
architecture v1 of m41 is
begin
z <= a when sel = “00” else
b when sel = “01” else
c when sel = “10” else
d ;
end v1 ;
```

The testbench for Quickvhdl is

```
-- Philips CPLD Applications
-- m41_tb.vhd
-- 17 oct 1995
library ieee ;
use ieee.std_logic_1164.all ;
entity testbench is end ;
architecture tb of testbench is
component m41
```

```
port (a: in std_logic_vector (5 downto 0);
b: in std_logic_vector (5 downto 0);
c: in std_logic_vector (5 downto 0);
d: in std_logic_vector (5 downto 0);
sel: in std_logic_vector (1 downto 0);
z: out std_logic_vector (5 downto 0));

end component ;
signal a : std_logic_vector (5 downto 0) ;
signal b : std_logic_vector (5 downto 0) ;
signal c : std_logic_vector (5 downto 0) ;
signal d : std_logic_vector (5 downto 0) ;
signal sel : std_logic_vector (1 downto 0) ;
signal z : std_logic_vector (5 downto 0) ;
signal vector_cnt : integer := 1 ;
signaltype test_record is record
a : std_logic_vector (5 downto 0) ;
b : std_logic_vector (5 downto 0) ;
c : std_logic_vector (5 downto 0) ;
d : std_logic_vector (5 downto 0) ;
sel : std_logic_vector (1 downto 0) ;
z : std_logic_vector (5 downto 0) ;
end record ;
type test_array is array(positive range<>) of test_record ;
constant test_vectors : test_array := (
-- a, b, c, d, sel, z
(“000000”,  “000111”, “111000”, “111111”, “00”, “000000”),
(“000000”,  “000111”, “111000”, “111111”, “01”, “000111”),
(“000000”,  “000111”, “111000”, “111111”, “10”, “111000”),
(“000000”,  “000111”, “111000”, “111111”, “11”, “111111”)
);
begin
dut: m41 port map (a => a,
b => b,
c => c,
d => d,
sel => sel,
z => z);
testrun:  process
variable vector : test_record ;
begin
for index in test_vectors’range loop
vector_cnt <= index ;
vector := test_vectors(index);
a <= vector.a ;
b <= vector.b ;
c <= vector.c ;
d <= vector.d ;
sel <= vector.sel ;
wait for 50 ns ;
if ( z /= vector.z) then
error_flag <= ‘1’ ;
assert false
report “Output did not match.” ;
else
error_flag <= ‘0’ ;
end if ;
end loop ;
wait ;
end process ;
end;
```

To begin the design flow, from the project directory enter

```
mkdir src
cp <path>/m41.vhd src
cp <path>/m41_tb.vhd src
```

Then create and map the llibrary

```
qvlib work
qvmap work “<path>/work”
```

Now compile the files

```
qvcom src/m41.vhd -work work -synth

qvcom src/m41_tb.vhd -work work -synth
```

Now start autologic for synthesis by entering alui -nodisplay.

```
/export/home/lss/designs/mgc/vhdl/practice3->alui -nodisplay
```

This will cause the following to be displayed. The Autologic session is given in the next three pages. A prompt precedes user input. Lines without a prompt are Autologic output.

```
AutoLogic II Optimizer v8.4_3.2  Tue Jun 27 10:34:53 PDT 1995
Autologic idea license granted
GENIE version 9.16
Loading library -- /export/home2/mgc/pkgs/se_any/userware/default/ipc.ma
Loading library -- /export/home2/mgc/pkgs/syn_any/userware/default/autologic.ma
Loading AutoLogic Timing Driven Layout Library.
Loading library -- /export/home2/mgc/pkgs/syn_any/userware/default/opt_cli.ma
Loading library -- /export/home2/mgc/pkgs/syn_any/userware/default/complib.ma
Loading library -- /export/home2/mgc/pkgs/np_any/userware/default/gc_util.ma
Loading file -- /export/home2/mgc/pkgs/syn_any/userware/default/lo.m
loading /export/home2/mgc/pkgs/np_any/userware/default/synrc.m

   Copyright (c) Mentor Graphics Corporation, 1990-1995
              All Rights Reserved.
          UNPUBLISHED, LICENSED SOFTWARE
      CONFIDENTIAL AND PROPRIETARY INFORMATION
 PROPERTY OF MENTOR GRAPHICS CORPORATION OR ITS LICENSORS

Loading Generic Technology
loading /export/home2/mgc/pkgs/aui_any/userware/default/opt.m
Redefinition of function update_all_vhdl_library_browser
Redefinition of function env_INI
Redefinition of function read_vhdl
>
> opn design -vhdl src/m41.vhd
# opn design -vhdl src/m41.vhd
```

```
alx-hdl - v8.4_1.22
Messages will be logged to file '/export/home2/mgc/tmp/gn842537606.log'
-- Reading file /export/home2/mgc/pkgs/qvhdl_libs/data/standard.vhd
-- Loading package STANDARD into library STD
-- Reading root vhdl file src/m41.vhd
-- Reading file /export/home2/mgc/pkgs/qvhdl_libs/data/std_1164.vhd
-- Loading package STD_LOGIC_1164 into library IEEE
-- Loading entity M41 into library MGC_WORK
-- Loading architecture V1 of M41 into library MGC_WORK
-- Compiling root entity M41(V1)
-- VHDL source successfully analyzed

##  Loading $MGC_HOME/lib/autologic.ini
Warning: Overwriting netlist mgc_operators.eq_2u_2u(INTERFACE)
Warning: Overwriting netlist mgc_work.m41(v1)
0
> > env dst gen_lib
# env dst gen_lib
Loading Technology -- /export/home2/lib/synlib/gen_lib
*** gen_lib Library - Version 1.3 - 01Nov93
Loading Cell Definition -- bin/celldef.ma
 Loading Database -- /export/home2/lib/synlib/gen_lib/celldb
Warning: Cannot find auxiliary rules
        Consider creation of auxiliary rules using "do_aux_rules <libname>"
0
> syn vhdl work m41 -arch v1
Synthesizing ....
alx-hdl - v8.4_1.22
Messages will be logged to file '/export/home2/mgc/tmp/synth842537685.log'
-- Reading file /export/home2/mgc/pkgs/qvhdl_libs/data/standard.vhd
-- Loading package STANDARD into library STD
-- Reading file /export/home/lss/designs/mgc/vhdl/practice3/work/m41/m41__alx.vhd into library work
-- Reading file /export/home2/mgc/pkgs/qvhdl_libs/data/std_1164.vhd
-- Loading package STD_LOGIC_1164 into library IEEE
-- Loading entity M41 into library work
-- Reading root vhdl file /export/home/lss/designs/mgc/vhdl/practice3/work/m41/v1__arch__alx.vhd
-- Loading architecture V1 of M41 into library work
-- Compiling root entity M41(V1)
-- VHDL source successfully analyzed

##  Loading $MGC_HOME/lib/autologic.ini
Warning: Overwriting netlist mgc_operators.eq_2u_2u(INTERFACE)
Warning: Overwriting netlist work.m41(v1)
netlist:work:m41.v1

> opt area -low
> opt area -low
# opt area -low
DMAG library version 1.0 installed
   Creating new view (mgc_operators eq_2u_2u_Zd6dc662 INTERFACE)
   Creating new view (mgc_operators eq_2u_2u_Z1fb8c43 INTERFACE)
Starting top-level cell: m41 v1

Optimizing /v1
Netlist Count 1 of 1
Starting area optimization, effort:  low, factoring option:  factor
         original area  litweight     current area  litweight
    top:            0        890 ==>           0        890
Starting combinational optimization
         original area  litweight     current area  litweight
    top:            0        890 ==>           0        890
  local:            0        570 ==>           0        570
End of combinational optimization
         original area  litweight     current area  litweight
    top:            0        890 ==>       34000        320
  local:            0        570 ==>       34000          0
End of area optimization, effort low
         original area  litweight     current area  litweight
    top:            0        890 ==>       34000        320
0
```

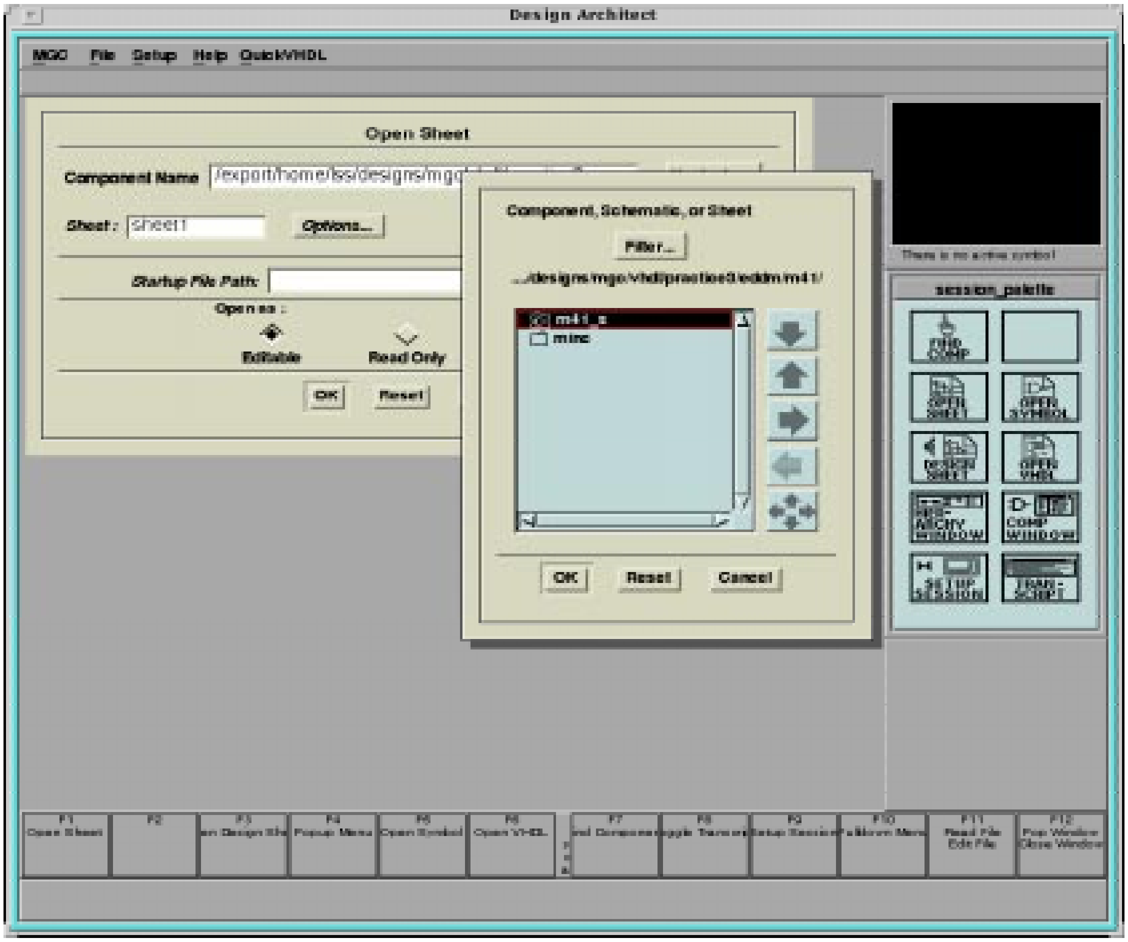
```
>sav design -eddm -model m41 -map eddm -schematic
```

This step produces alot of output and indicates that a schematic has been written.

```
> quit -f
```

Invoke Design Architect steps by entering

```
da &
```



Component Name: /export/home/lss/designs/mgc

Sheet: sheet1

Startup File Path:

Open as: ◆ Editable ◇ Read Only

.../designs/mgc/vhdl/practice3/eddm/m41/

m41_s
mirc

Figure 1. Opening a schematic in Design Architect..

Open the schemaitc by selecting Open Sheet in the palette window and selecting m41_s as shown.

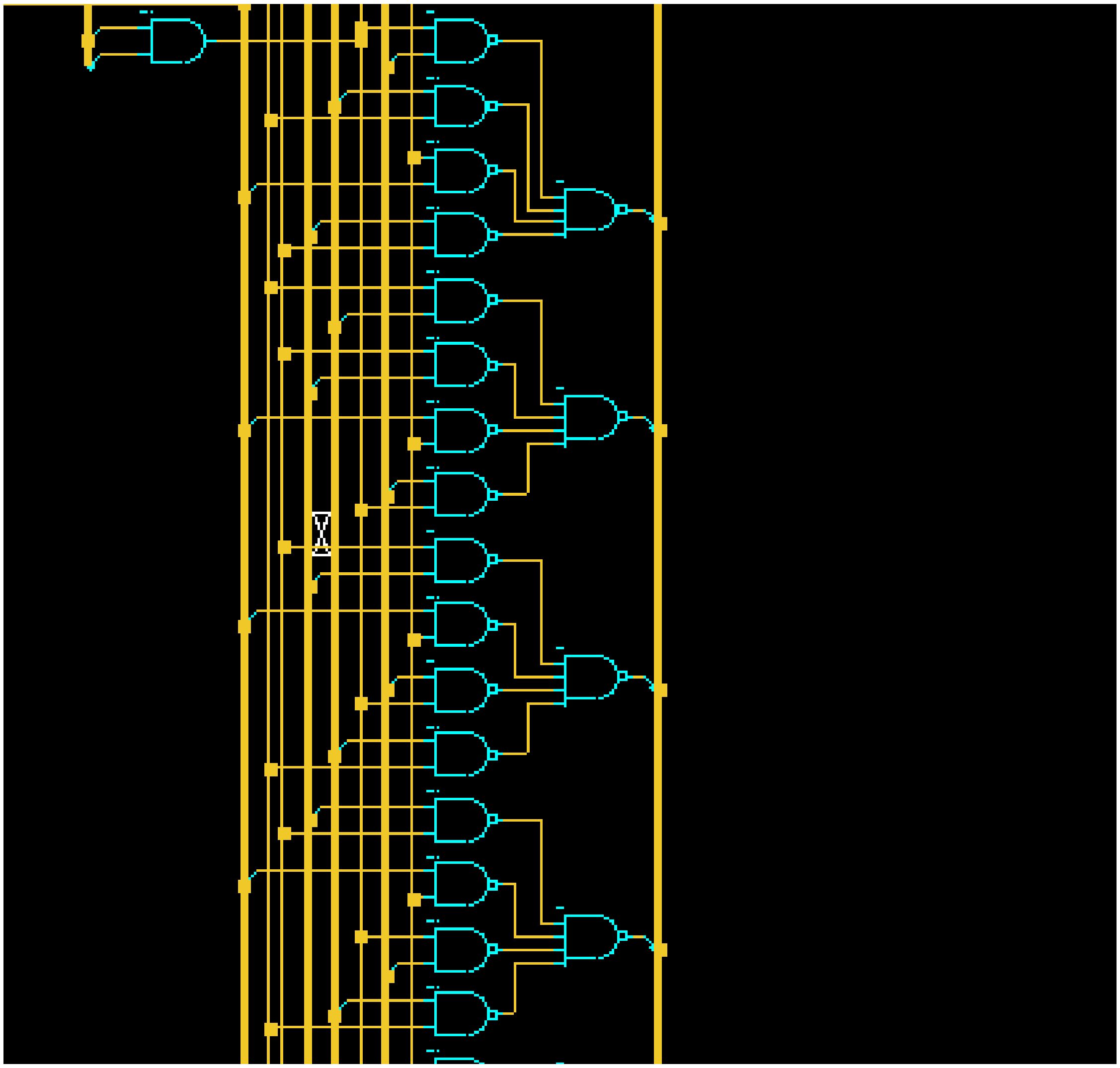
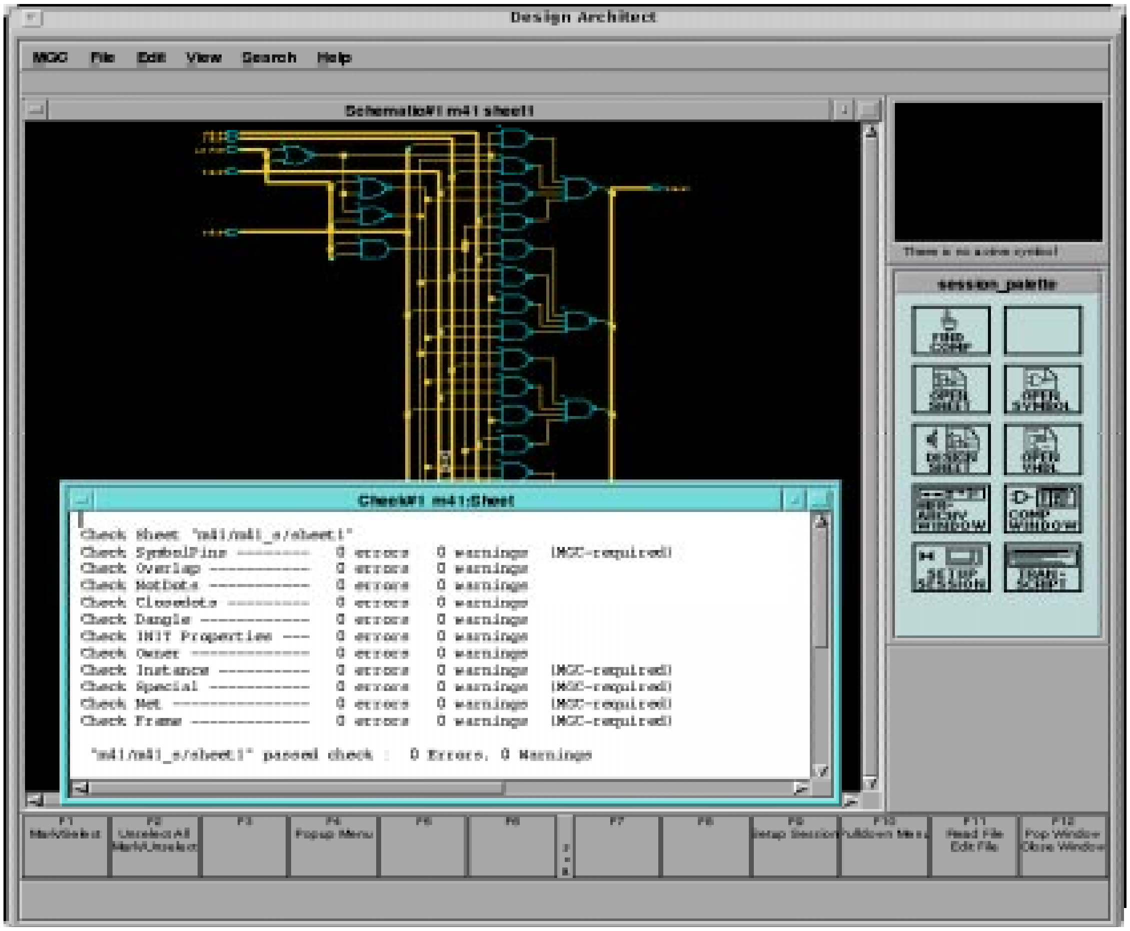


Figure 2. m41_s schematic

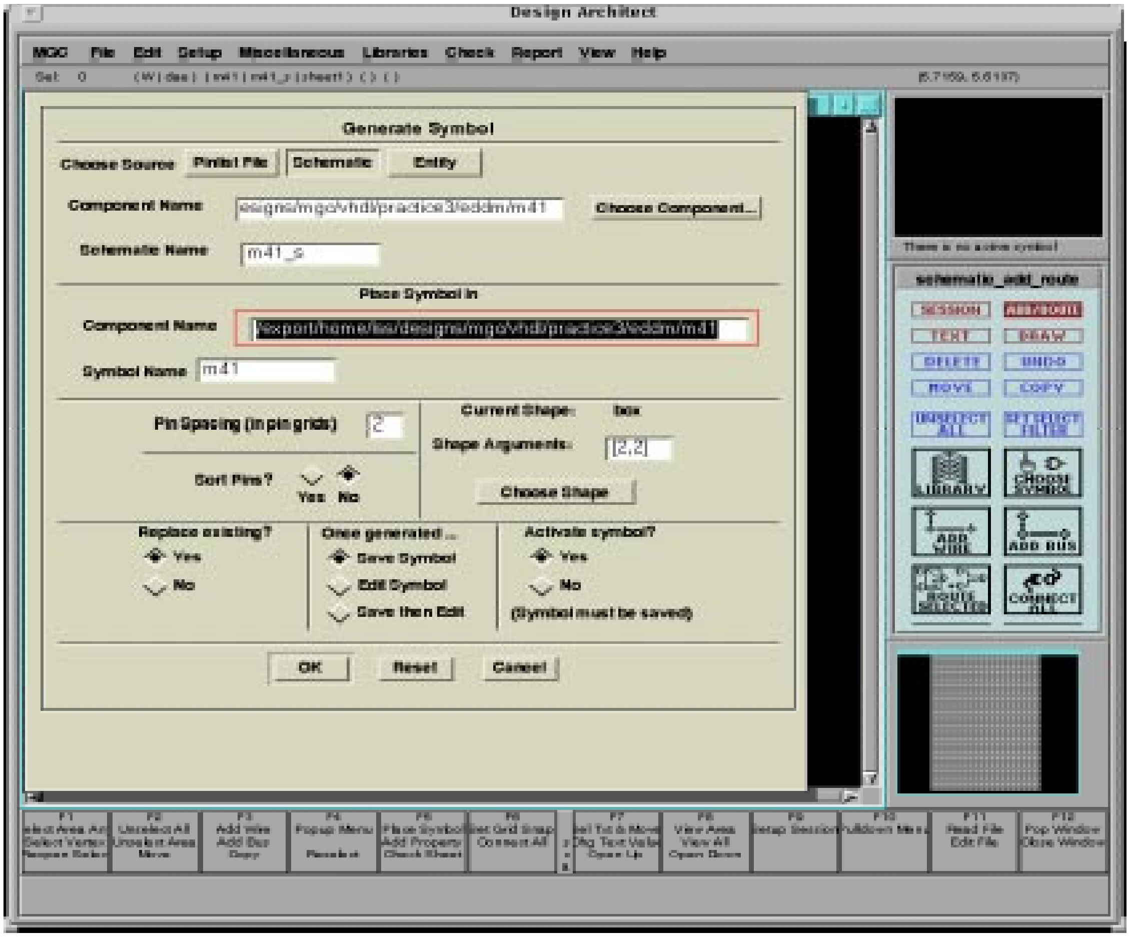


```
Check Sheet 'm41/m41_s/sheet1'
Check SymbolPins ---------   0 errors   0 warnings   (MGC-required)
Check Overlap ------------   0 errors   0 warnings
Check NotDots ------------   0 errors   0 warnings
Check Closedots ----------   0 errors   0 warnings
Check Dangle -------------   0 errors   0 warnings
Check INIT Properties ---   0 errors   0 warnings
Check Owner --------------   0 errors   0 warnings
Check Instance -----------   0 errors   0 warnings   (MGC-required)
Check Special ------------   0 errors   0 warnings   (MGC-required)
Check Net ----------------   0 errors   0 warnings   (MGC-required)
Check Frame --------------   0 errors   0 warnings   (MGC-required)

 'm41/m41_s/sheet1' passed check :  0 Errors, 0 Warnings
```

Figure 3. Electrical rules check results

With the schematic open, use the pull down menus from the Menu bar Check to ensure that there are no electrical rule violations.



Generate Symbol

Choose Source: Pinlist File | [✓ Schematic] | Entity

Component Name: esigns/mgo/vhdl/practice3/eddm/m41

Schematic Name: m41_s

Place Symbol in

Component Name: /export/home/lie/designs/mgo/vhdl/practice3/eddm/m41

Symbol Name: m41

Pin Spacing (in pin grids): 2

Sort Pins? ○ Yes ● No

Current Shape: box

Shape Arguments: [2,2]

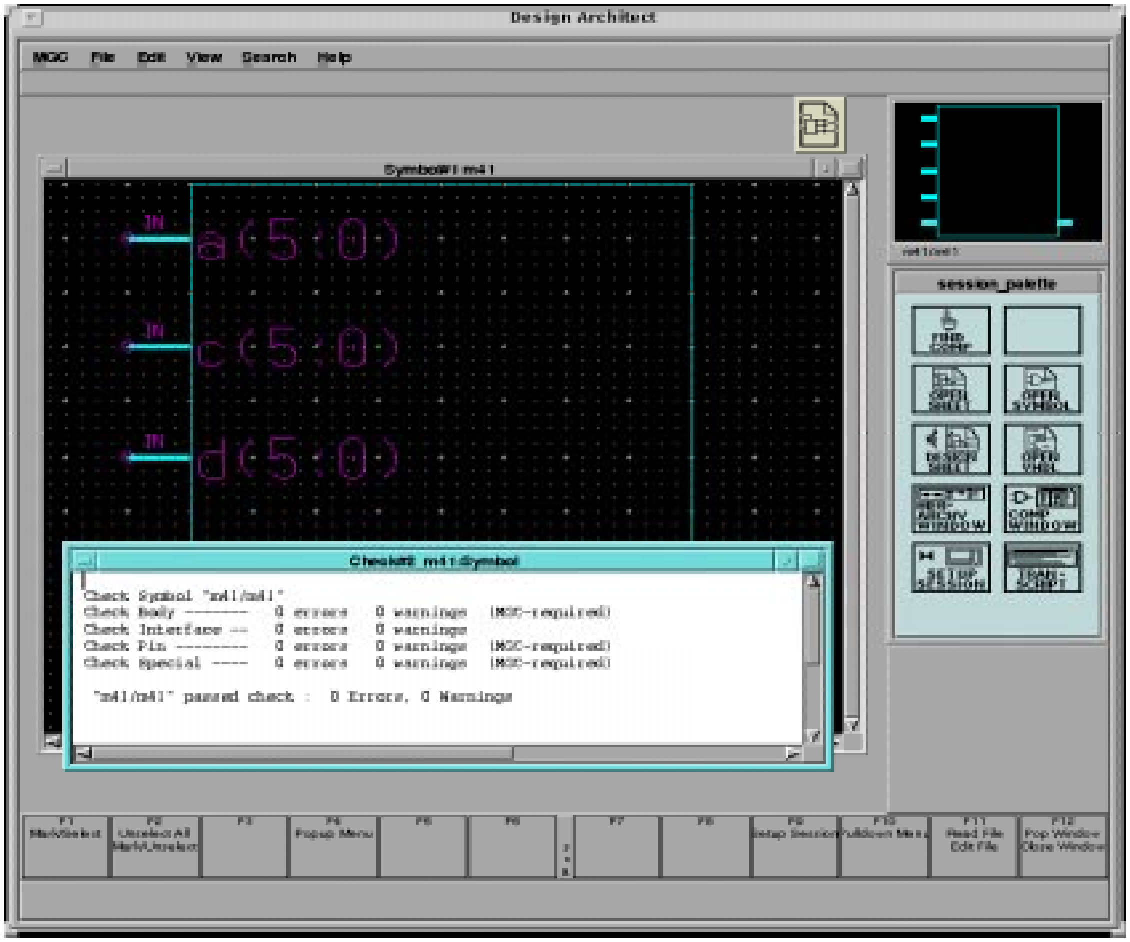
Replace existing? ● Yes ○ No

Once generated ... ● Save Symbol ○ Edit Symbol ○ Save then Edit

Activate symbol? ● Yes ○ No (Symbol must be saved)

Figure 4. Generating a symbol

After the schematic is checked without errors, use the Menu bar to select Miscellaneous-Generate Symbol, and change the radio buttons in the dialog box to Replace Existing, Save, and Activare the symbol.

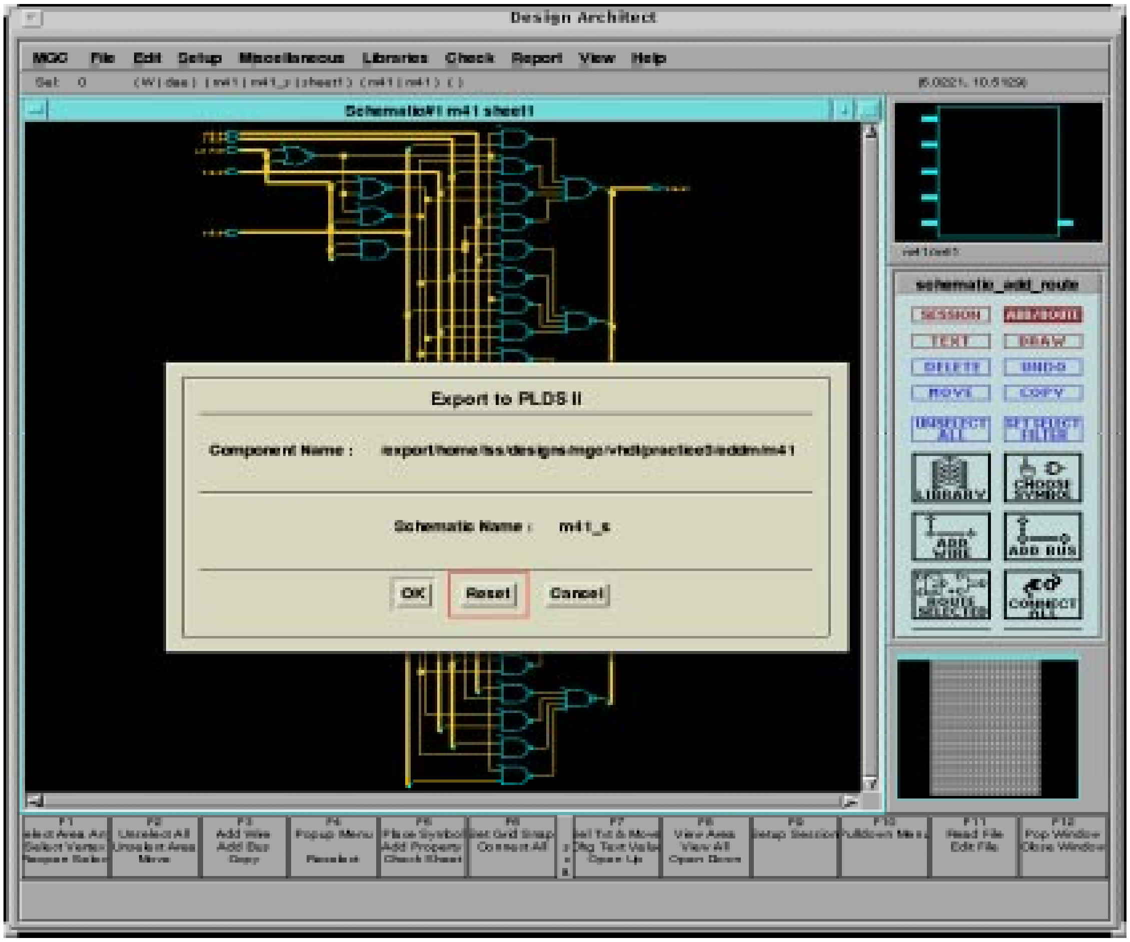


```
Check Symbol "m41/m41"
Check Body --------   0 errors   0 warnings  (MGC-required)
Check Interface --    0 errors   0 warnings
Check Pin ---------   0 errors   0 warnings  (MGC-required)
Check Special ----    0 errors   0 warnings  (MGC-required)

 "m41/m41" passed check :  0 Errors, 0 Warnings
```

Figure 5. Check the symbol

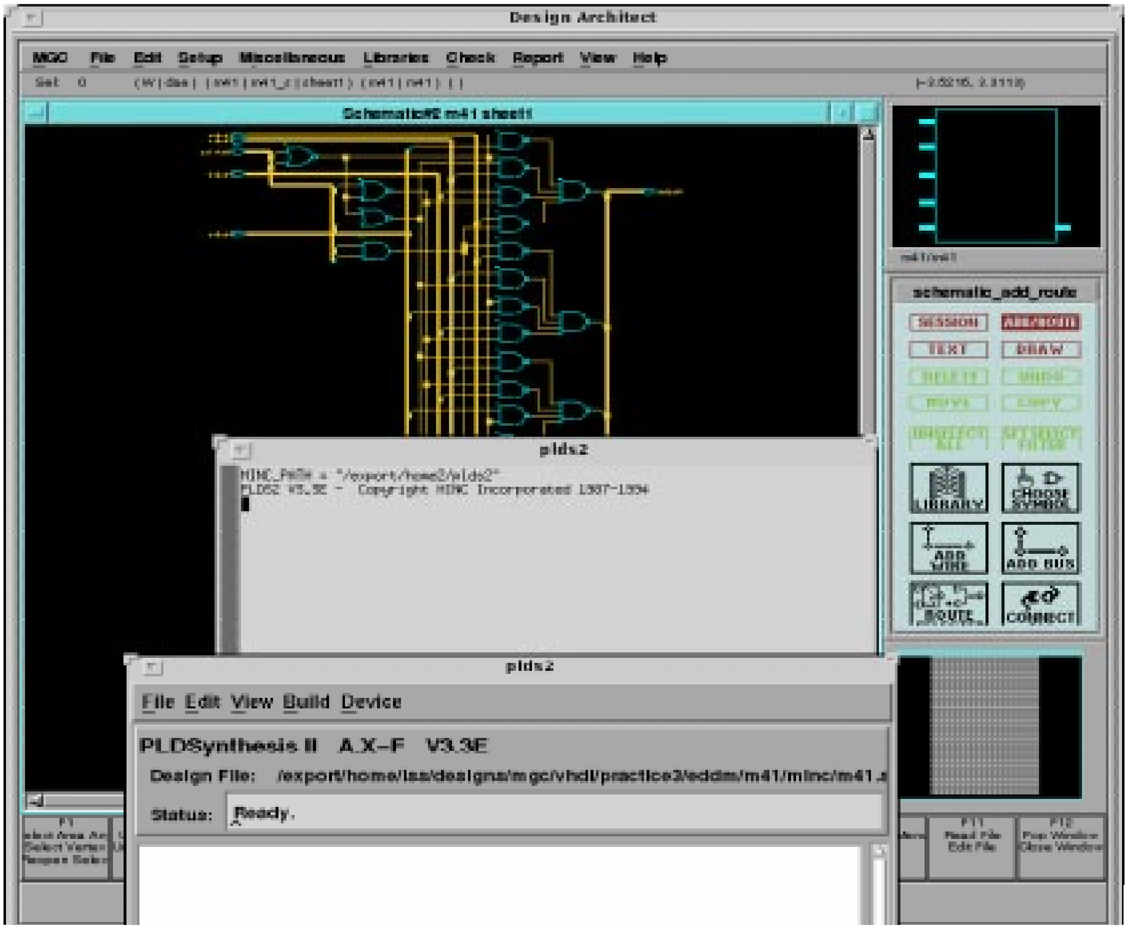
Close the schematic (upper right corner). Use the palette to Open Symbol. From the Menu bar, check the symbol with default registration.



Component Name : /export/home/lss/designs/mgc/vhdl/practice3/addm/m41

Schematic Name : m41_s

Figure 6. Export to PLDSII



```
MINC_PATH = "/export/home2/plds2"
PLDS2 V3.3E -  Copyright MINC Incorporated 1987-1994
```

PLDSynthesis II A.X–F V3.3E

Design File: /export/home/las/designs/mgc/vhdl/practice3/eddm/m41/minc/m41.s

Status: Ready.

Figure 7. Invoking plds2

This shows that PLDSII ha successfully written a.src file which can be compiled to a Philips CPLD. The PLDesigner GUI is shown in the lower part of the screen shot.

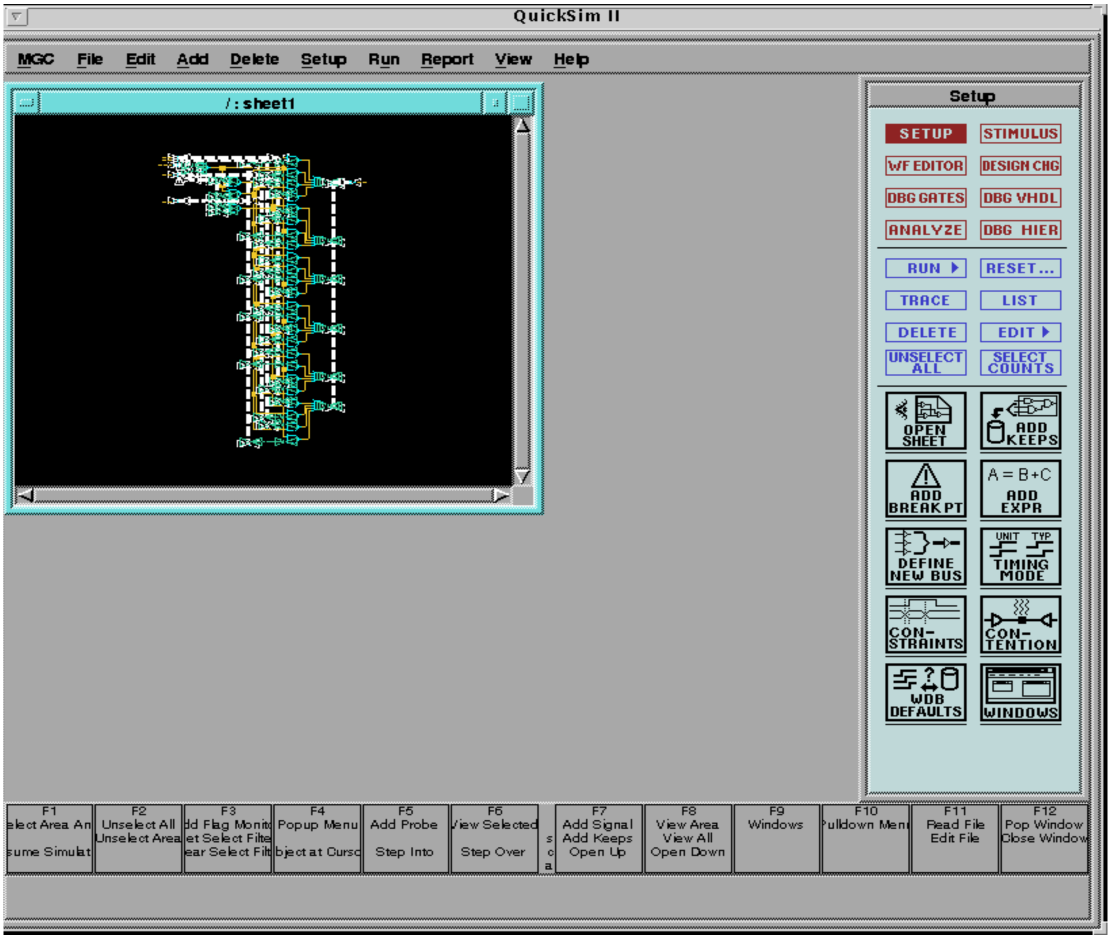
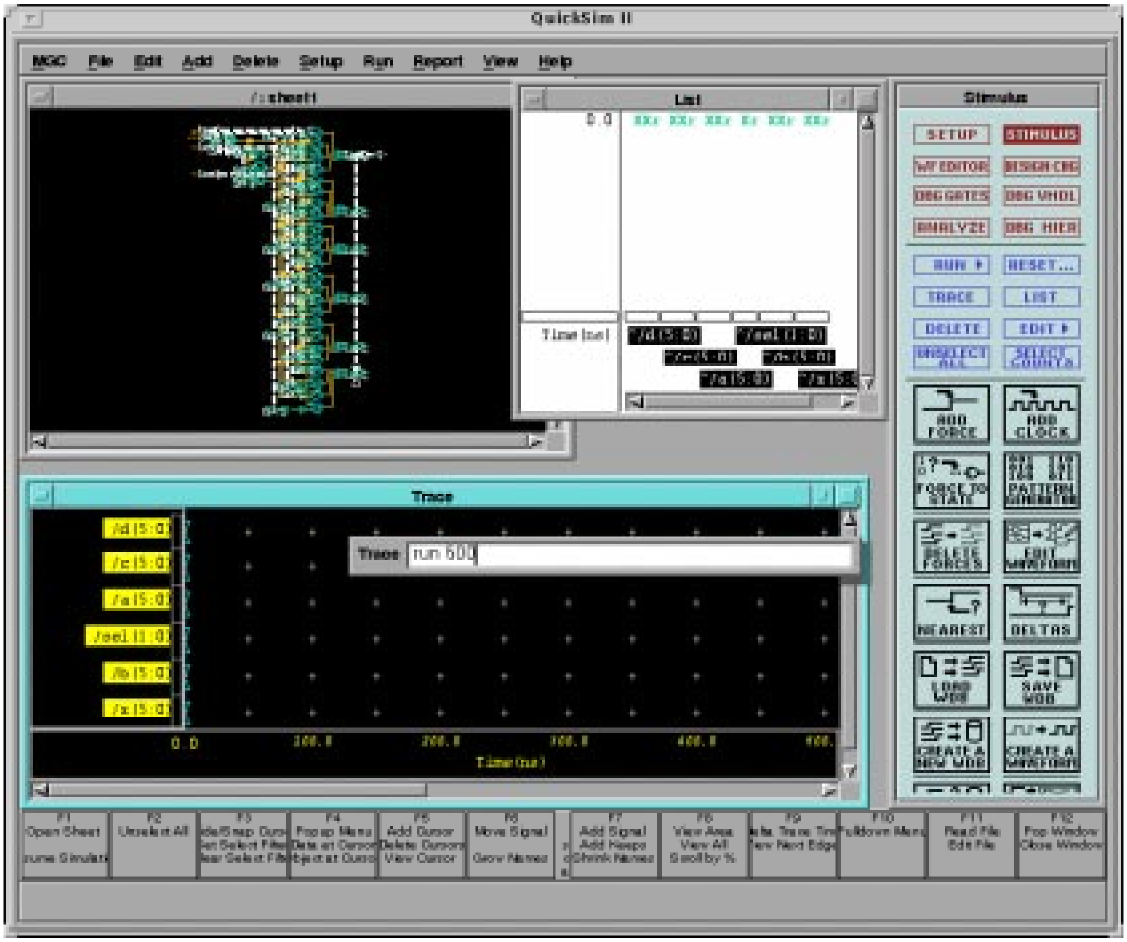


Figure 8. Using Quicksim II

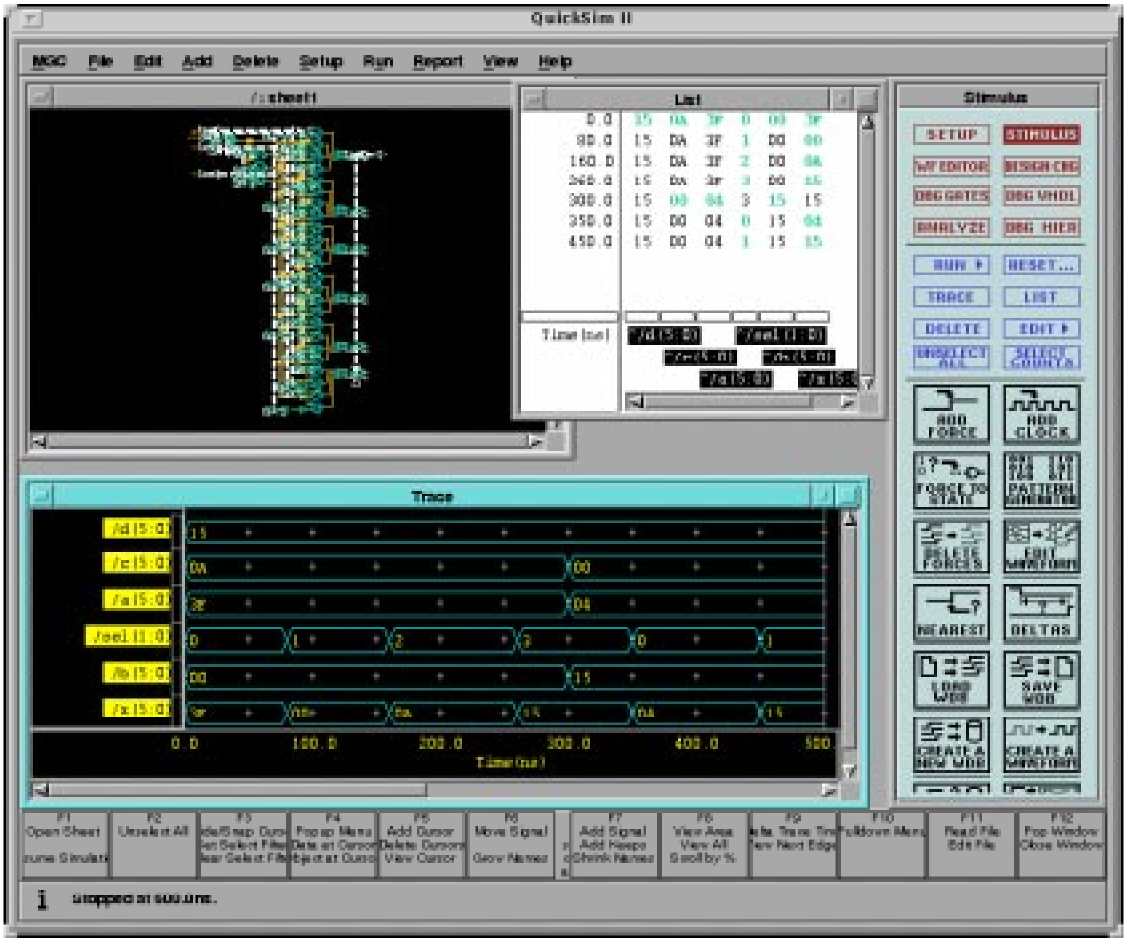
To simulate with Quicksim II, enter quicksim -timing_mode typ plds2_vpt. From the palette menu, select Open Sheet. Using the mouse, elect the nets to monitor in the schematic. Invoke Trace and List from the palette.



Trace run 500

Figure 9. Schematic, Trace, and List windows displayed.

To run a simulation, from the palette menu, enter Stimulus and provide the input stimuli. Enter run 500 to run a simulation.



```
     0.0   15  DA  3F  0  DD  3F
    80.0   15  DA  3F  1  DD  DD
   160.0   15  DA  3F  2  DD  DA
   260.0   15  DA  3F  3  DD  15
   300.0   15  DD  04  3  15  15
   350.0   15  DD  04  0  15  04
   450.0   15  DD  04  1  15  15

Time (ns)  /d(5:0)      /sel(1:0)
               /c(5:0)      /b(5:0)
                   /a(5:0)      /x(5:0)
```

Stopped at 500.0 ns.

Figure 10. Simulation results

Mentor Graphics Design Flow for targeting Philips CPLDs

AN059

Definitions

Short-form specification — The data in a short-form specification is extracted from a full data sheet with the same type number and title. For detailed information see the relevant data sheet or data handbook.

Limiting values definition — Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.

Application information — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Disclaimers

Life support — These products are not designed for use in life support appliances, devices or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

Right to make changes — Philips Semiconductors reserves the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

Philips Semiconductors
811 East Arques Avenue
P.O. Box 3409
Sunnyvale, California 94088–3409
Telephone 800-234-7381

print code Date of release: 07-98

Document order number: 9397 750 04182