FPSLIC Training Series (FPGA based designs)

Software Requirements

- System Designer
  - -Exemplar – Leonardo Spectrum
  - -Modelsim Simulator
- CPS

Hardware Requirements

- ATSTK94 Starter Kit

SYSTEM DESIGNER™
Programmable System Level Integration Software
Copyright © 2001, Atmel Corporation, all rights reserved.
ATMEL®

LAB 1 - Synthesis with No Macro Generators andgate.vhd

LAB 2 - Synthesis with Automatic Macro Generators adder.vhd

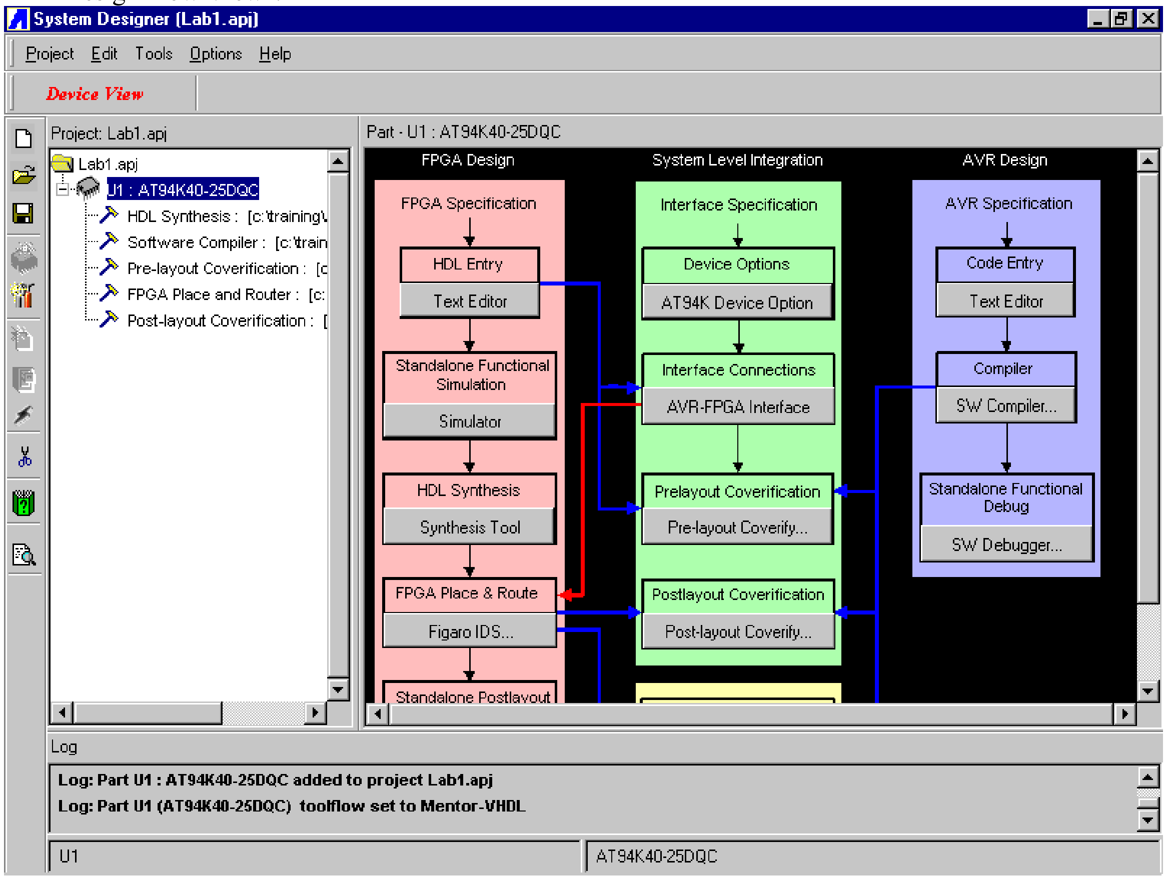
LAB 3 - Synthesis with Macros for Simulation Counter.vhd

LAB 4 - Synthesis with Black Box Macro Generators ramdesign.vhd

LAB 1 : LeonardoSpectrum flow with No Macro Generators

Creating a New Project in System Designer

1. Go to Start > Programs > Atmel > System Designer3.0 to open System Designer .
2. From the “Project” menu select “New…”
3. Select “New Project Wizard”. This will open the New Project Wizard, which allows you to choose your project directory, select which part you want to design with and configure the design tool flow.
4. Press “Next >”
5. Set-up the Design Directory – `C:\TRAINING\AVR\LAB1`
6. Setup the Design File Name – `LAB1`
7. Press “Next >”
8. In the “New Project Wizard Parts Dialog” select the AT94K40-25DQC, this is the 208-pin TQFP device on the ATSTK94.
9. Press “Next >”
10. In the “New Project Wizard Tool Flow Dialog select “Mentor – VHDL”
11. Press “Next >”
12. It is possible to add multiple parts to work on using the New Project Wizard Add More Parts Dialog, but for this lab session we will focus on a single AT94K device.
13. Press “Next >”
14. Press “Finish” on the New Project Wizard Congratulations Dialog.
15. Upon clicking the “Finish” button, the “Device View” will be present, click on the part to enter the “Design Flow View”.



```
Log: Part U1 : AT94K40-25DQC added to project Lab1.apj
Log: Part U1 (AT94K40-25DQC) toolflow set to Mentor-VHDL
```

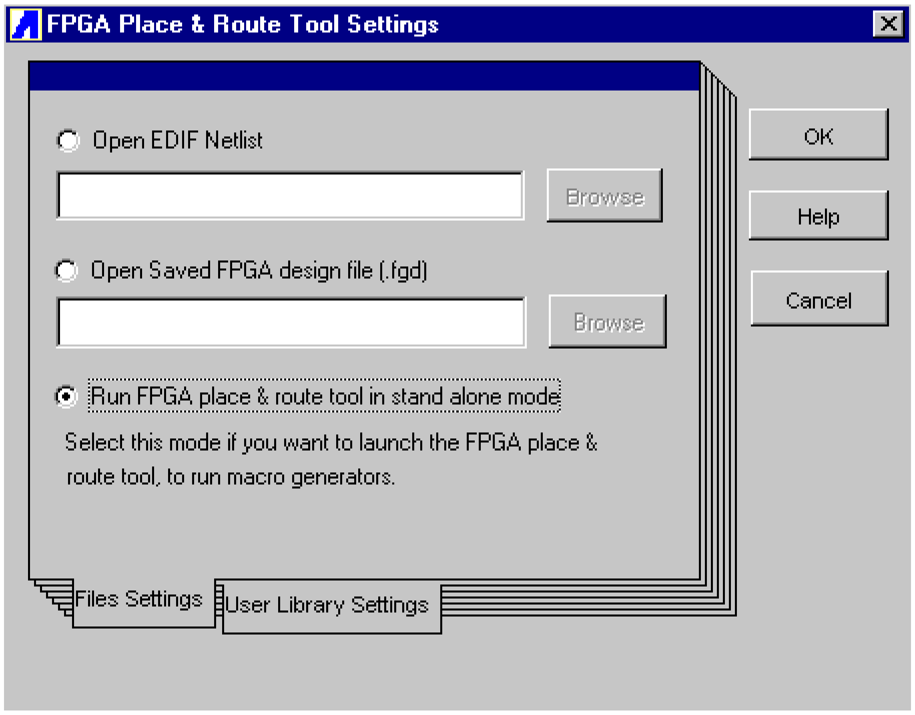
Figure 1: System Designer Design Flow

The Design Flow view shows the steps required to do a design for FPSLIC, using System Designer and coverification. The arrows on the diagram show dependencies between the steps. So for example if you want to run Pre-layout Coverification, you have to do HDL Entry, Code Entry and define the interface between the AVR and FPGA first.

In this lab we only concentrate on FPGA part of FPSLIC (FPGA Standalone) not on AVR/Coverification part of it. To do FPGA design entry we can either follow the pink path or use the HDL Planner with in FPGA Place and Route tool (**Figaro IDS**), we will choose the later part.

FPGA Place and Route (Figaro IDS) and HDL Planner

1. Press **Figaro IDS**, the FPGA Place and Route Tool Settings dialog box opens, see **Figure 01**.



FPGA Place & Route Tool Settings

( ) Open EDIF Netlist

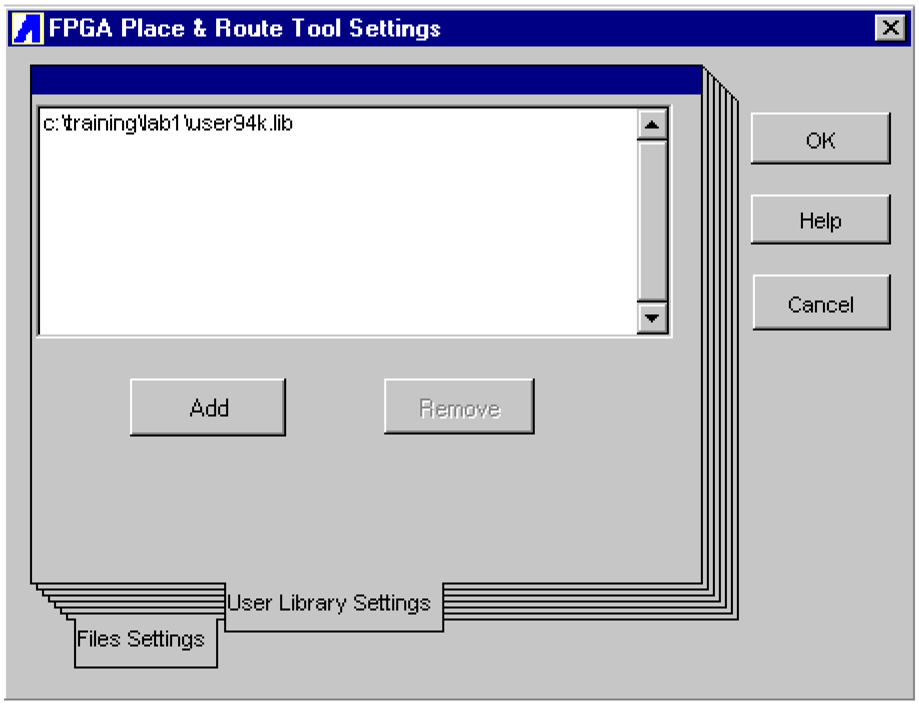
( ) Open Saved FPGA design file (.fgd)

(●) Run FPGA place & route tool in stand alone mode

Select this mode if you want to launch the FPGA place & route tool, to run macro generators.

Figure 01. FPGA Place and Route Tool Settings

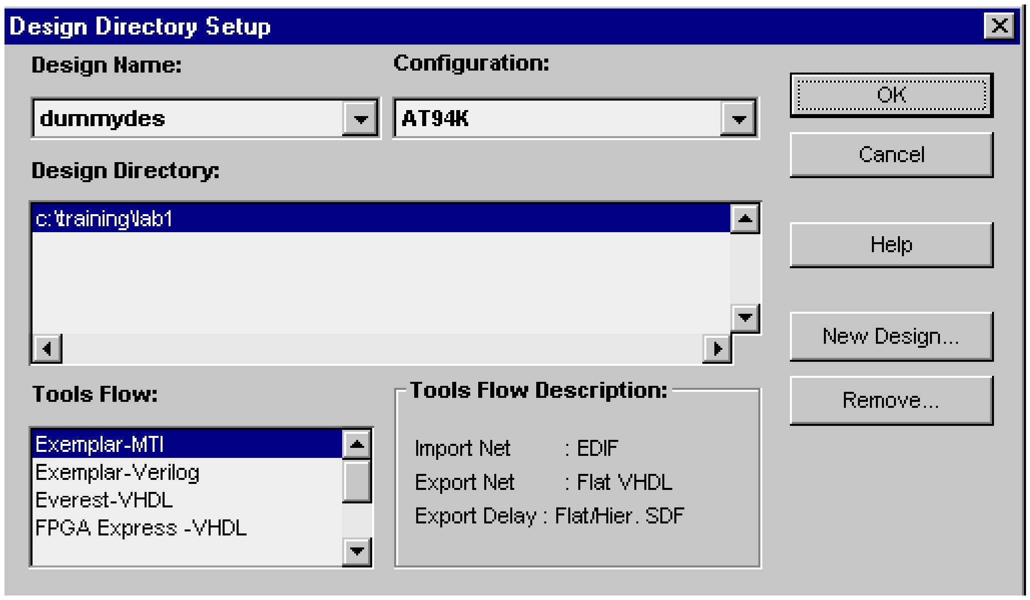
2. Select **Run FPGA place & route tool in stand alone mode**, another FPGA Place and Route Tool Settings dialog box appears, see **Figure 02**.



c:\training\lab1\user94k.lib

Figure 02. FPGA Place and Route Tool Settings

3. Take the default user library
4. Press **OK.** Figaro window pops up, follow the steps below to set up the design
5. Go to the **File** and choose **Design Setup**, see **Figure 03.**



Design Name: dummydes

Configuration: AT94K

Design Directory: c:\training\lab1

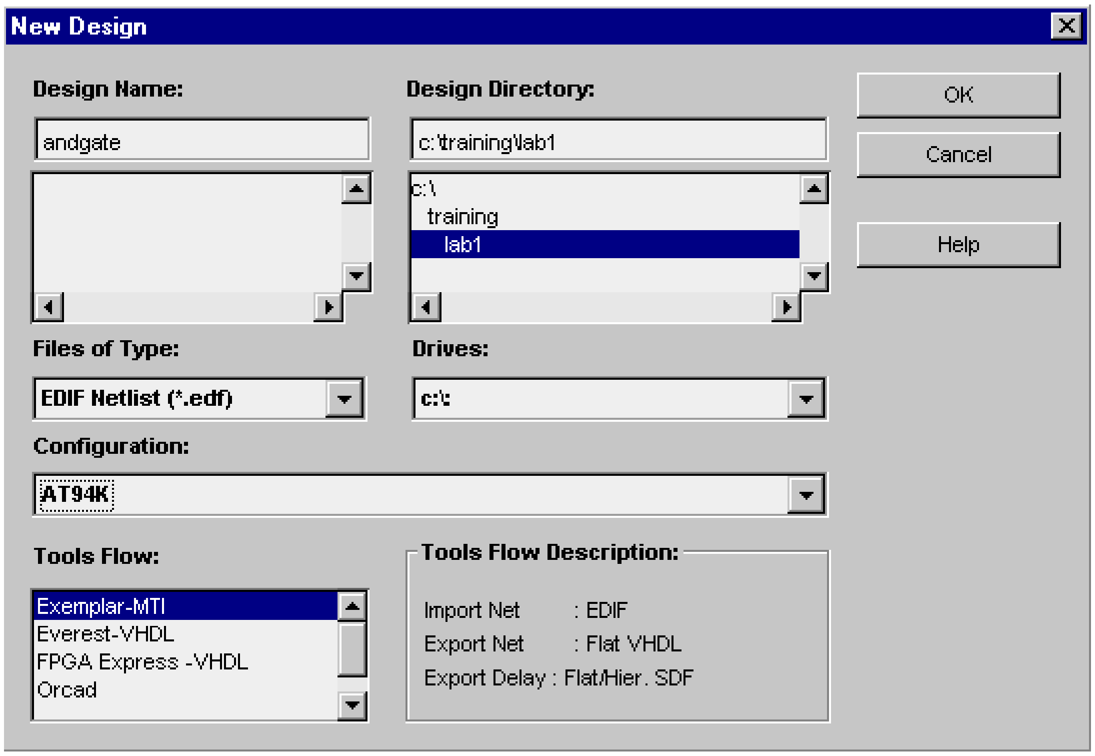
Tools Flow:
Exemplar-MTI
Exemplar-Verilog
Everest-VHDL
FPGA Express -VHDL

Tools Flow Description:
Import Net : EDIF
Export Net : Flat VHDL
Export Delay : Flat/Hier. SDF

Figure 03. Design Directory Setup

6. Press **New Design…**

7. Complete the fields as shown in **Figure 04**.


New Design

Design Name: andgate

Design Directory: c:\training\lab1

c:\
training
lab1

Files of Type: EDIF Netlist (*.edf)

Drives: c:\

Configuration: AT94K

Tools Flow:
Exemplar-MTI
Everest-VHDL
FPGA Express -VHDL
Orcad

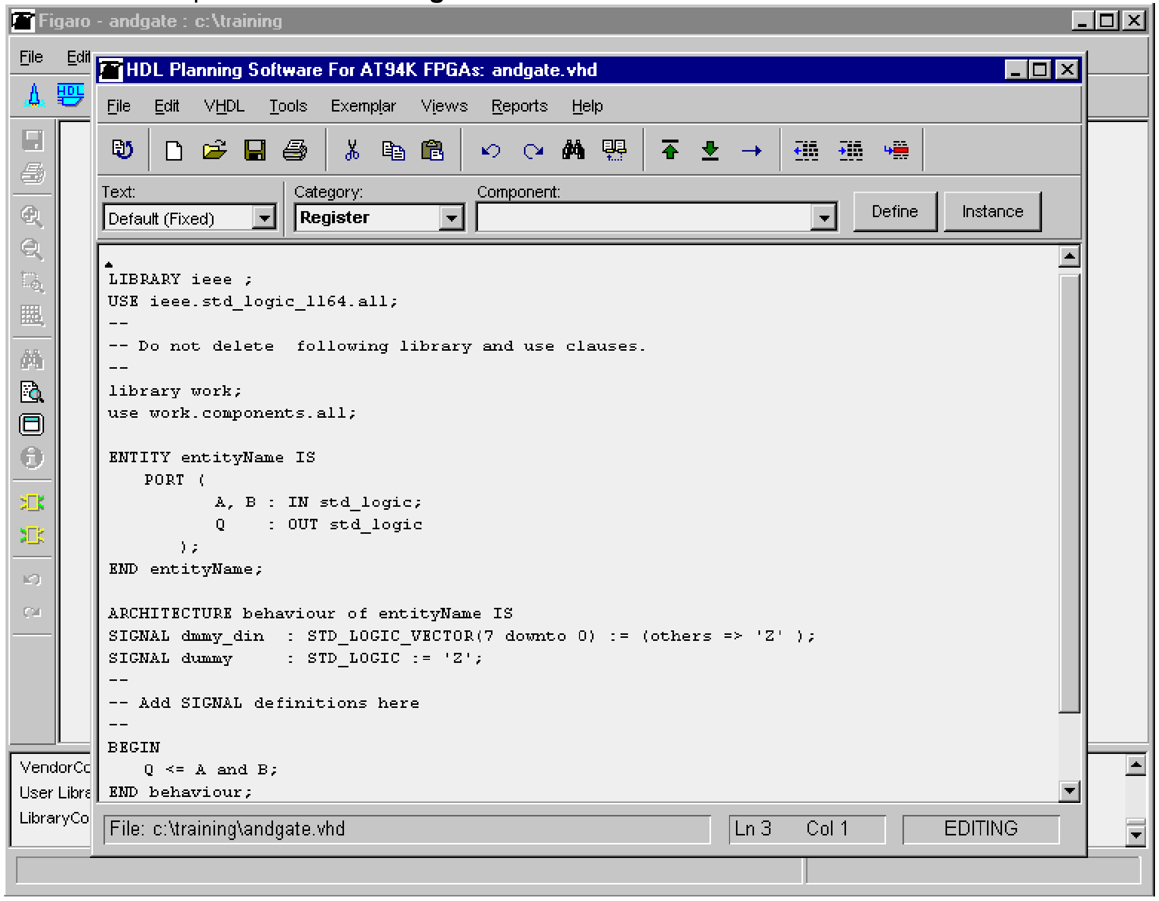
Tools Flow Description:
Import Net : EDIF
Export Net : Flat VHDL
Export Delay : Flat/Hier. SDF

Figure 04 New Design

8. Press **OK**

HDL Planner

1. Double-click on the **HDL Planner** icon or go to **Tools** → **HDLPlanner** in the figaro window to open the HDL planner.
2. Edit the template as shown in **Figure 05**.


Figaro - andgate : c:\training
HDL Planning Software For AT94K FPGAs: andgate.vhd
File Edit VHDL Tools Exemplar Views Reports Help
Text: Default (Fixed)
Category: Register
Component:
Define
Instance
VendorCo
User Libra
LibraryCo

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all;
--
-- Do not delete  following library and use clauses.
--
library work;
use work.components.all;

ENTITY entityName IS
    PORT (
            A, B : IN std_logic;
            Q    : OUT std_logic
        );
END entityName;

ARCHITECTURE behaviour of entityName IS
SIGNAL dmmy_din   : STD_LOGIC_VECTOR(7 downto 0) := (others => 'Z' );
SIGNAL dummy      : STD_LOGIC := 'Z';
--
-- Add SIGNAL definitions here
--
BEGIN
    Q <= A and B;
END behaviour;
```

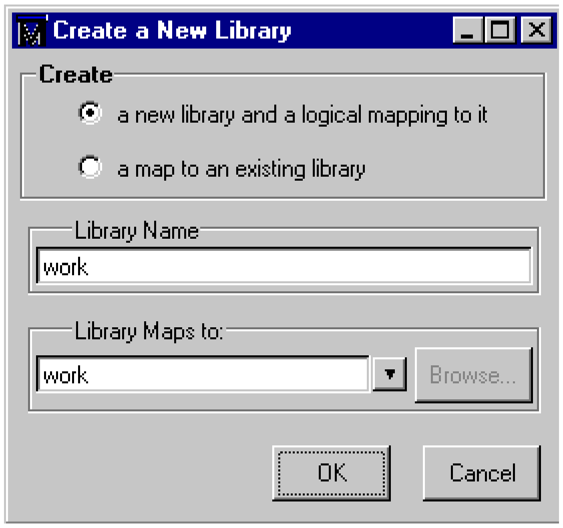
```
File: c:\training\andgate.vhd        Ln 3    Col 1        EDITING
```

Figure 05 HDLPlanner

3. Go to the **File** menu and select **Save As**. Name your project `andgate.vhd` in the directory created above.
4. Close HDL Planner
5. Close Figaro.

Functional Simulation using Modelsim simulator for Lab1

1. Press the **Simulation** button. ModelSim opens
2. Go to the **Design** menu and choose **Create a New Library**, a dialog box appears, see **Figure 06**.



Create: [✓ a new library and a logical mapping to it] a map to an existing library

Library Name: work

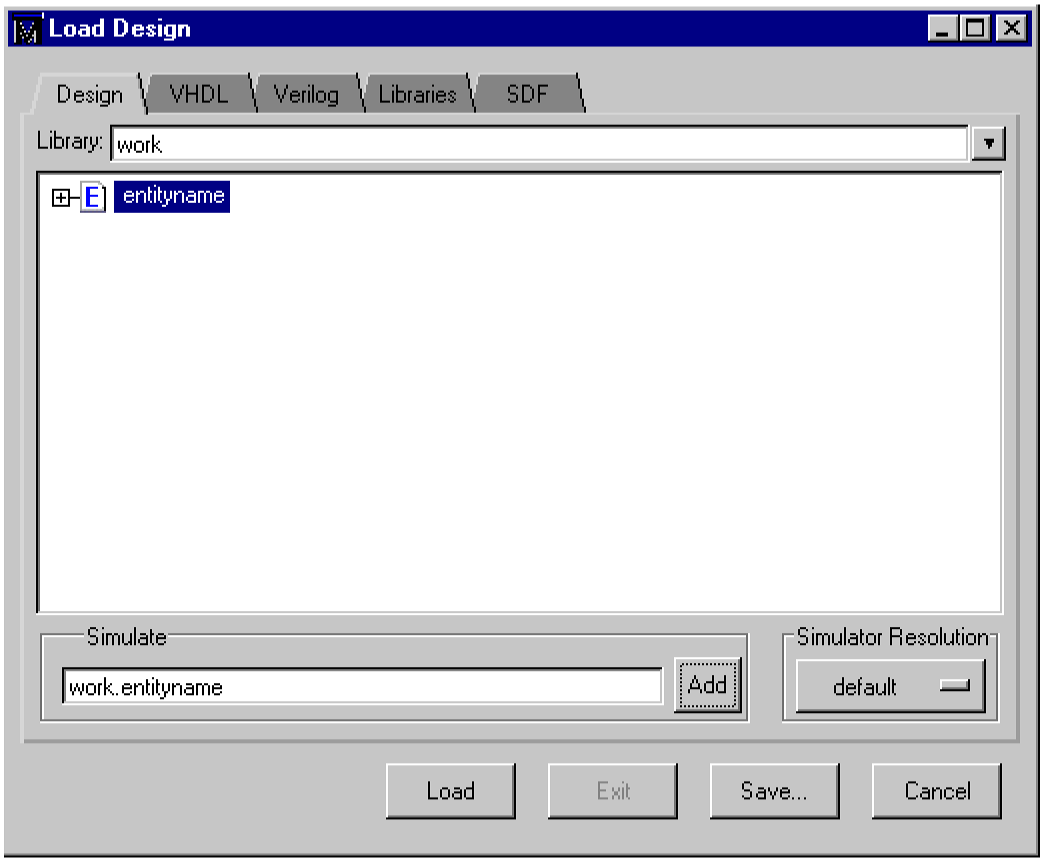
Library Maps to: work

Figure 06. New Library

3. Take the default and press OK
4. Go to the **Design** menu and choose **Compile**, a browser window appears, select **andgate.vhd** and **by atmel.vhd**
5. Press **Compile**
6. Press **Done**

Note: *atmel.vhd* file contains the package information that was created by Figaro when andgate.vhd is saved in HDL planner

7. Follow next two steps to load the design
8. Go to the **Design** menu and choose **Load Design**, the Load Design window appears, see **Figure 07**.



Library: work

entityname

Simulate: work.entityname

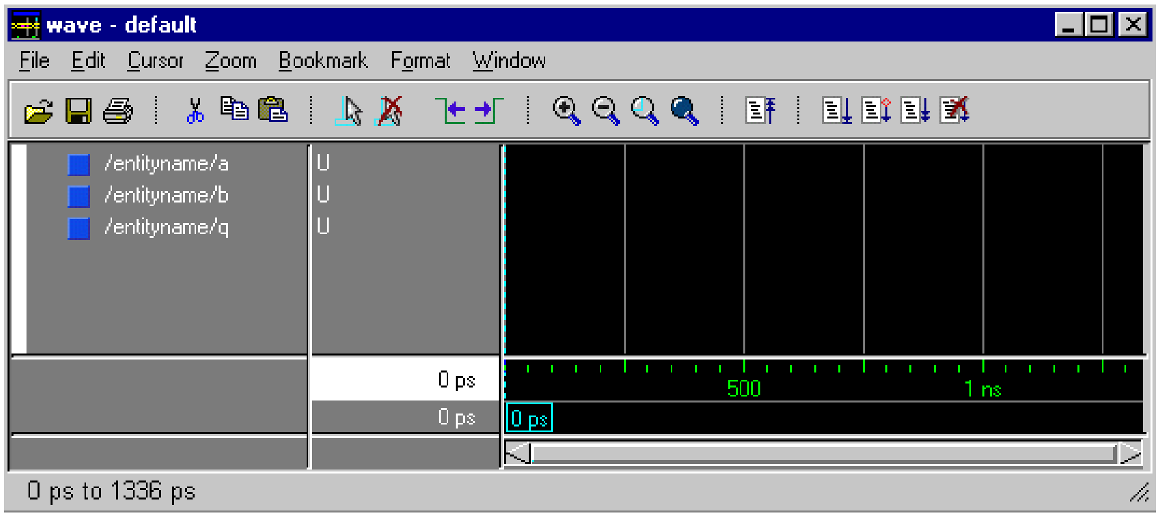
Simulator Resolution: default

Figure 07. Load Design

9. Select **entityname**
10. Press **Add**
11. Press **Load.** Ignore the following comments:

    #Cannot change directory while a simulatiion is in progress.
    #Use the “quit –sim” command to unload the design first.

12. Type **add wave –r /*** to open the wave window, see **Figure 08.**



```
/entityname/a   U
/entityname/b   U
/entityname/q   U
0 ps
0 ps
0 ps to 1336 ps
```

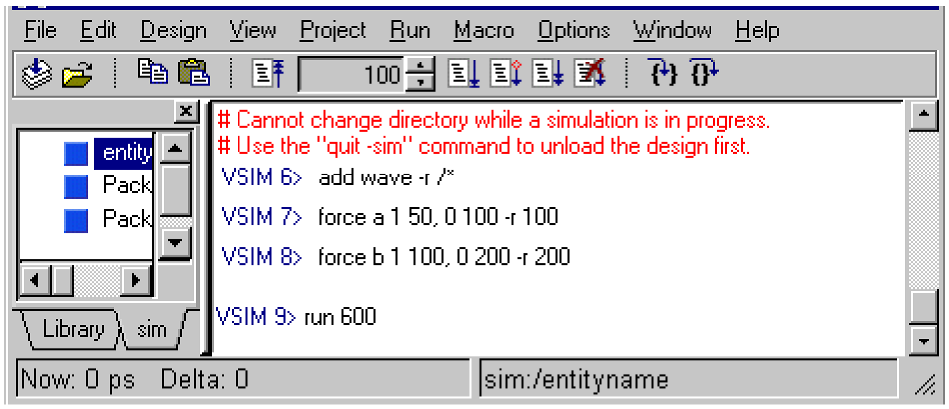
Figure 08. Wave

13. Apply the following stimulus in the Modelsim window, see **Figure 09**.

```
force a 1 50, 0 100 -r 100
force b 1 100, 0 200 -r 200
run 600
```

The force command forces “a” to value 1at 50 ns after the current time; then to “0” 100 ns from now; and repeat this cycle for every 100 ns

The run command will simulate until 600 ns



```
# Cannot change directory while a simulation is in progress.
# Use the "quit -sim" command to unload the design first.
VSIM 6> add wave -r /*
VSIM 7> force a 1 50, 0 100 -r 100
VSIM 8> force b 1 100, 0 200 -r 200

VSIM 9> run 600
```

```
Now: 0 ps   Delta: 0      sim:/entityname
```

Figure 09 Stimulus

14. Watch wave window for the expected results, see **Figure 10**.

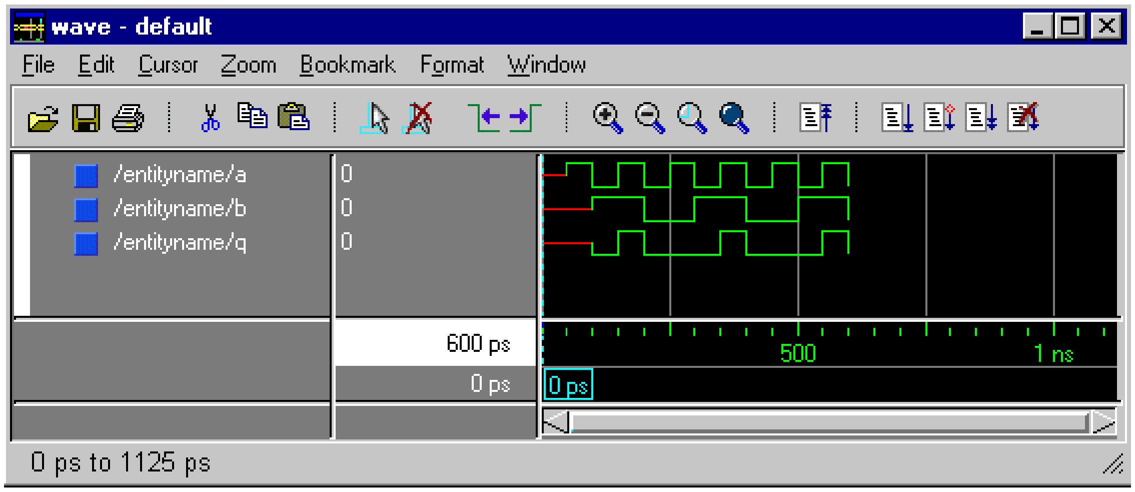
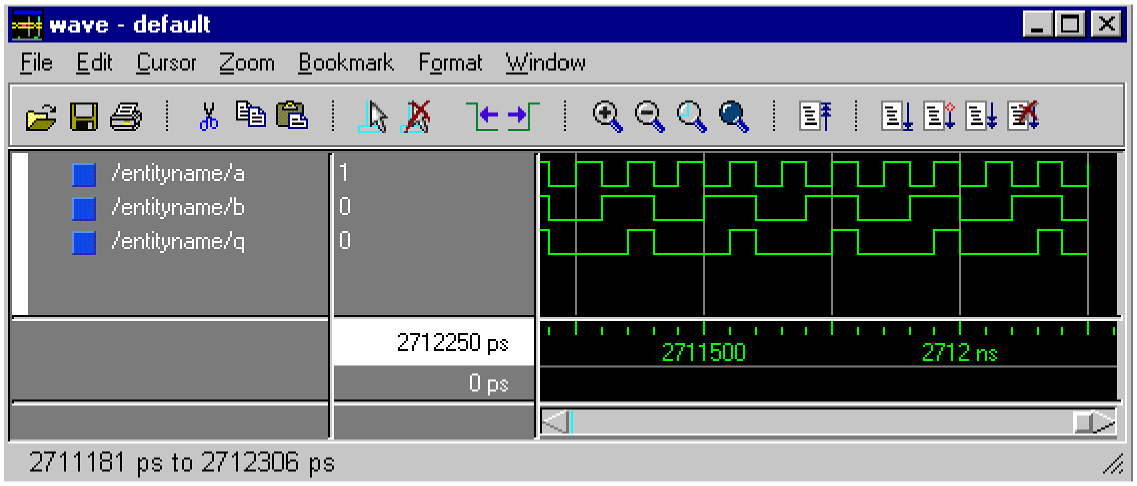


Figure 10 Wave

15. Go to the **Run** menu and select **Run –All**.

16. Press the **Break** button after few seconds to stop running. Watch wave window for the expected results, see **Figure 11**.



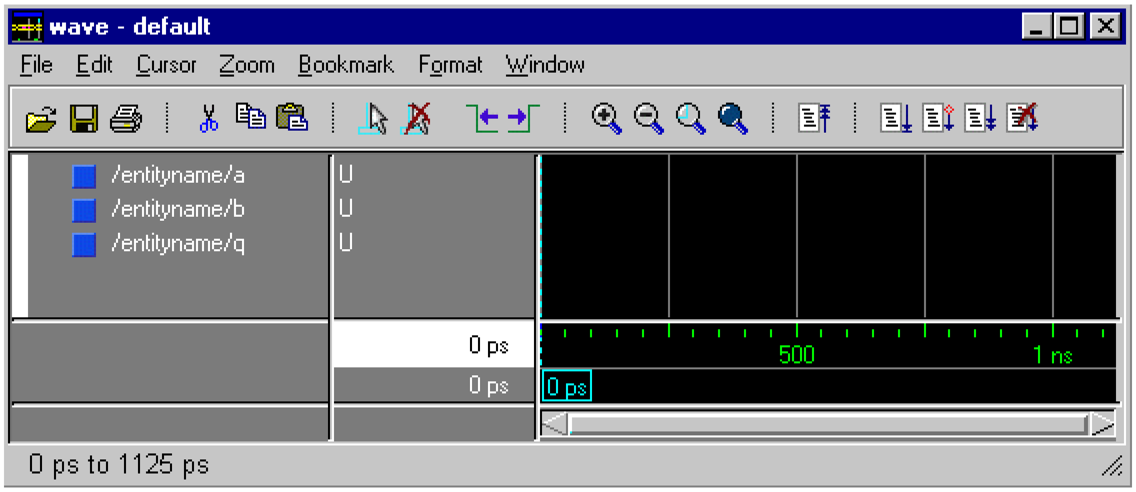
/entityname/a 1
/entityname/b 0
/entityname/q 0

2712250 ps
0 ps

2711181 ps to 2712306 ps

Figure 11 Wave

17. Go to the **Run** menu and select **Restart,** this clears the waveform window as shown in **Figure 12**.



/entityname/a U
/entityname/b U
/entityname/q U

0 ps
0 ps

0 ps to 1125 ps

Figure 12 Wave

The values can also be forced by:

1. Go to the **View** menu and choose **Signals**, the signals window appears, see **Figure 13**.

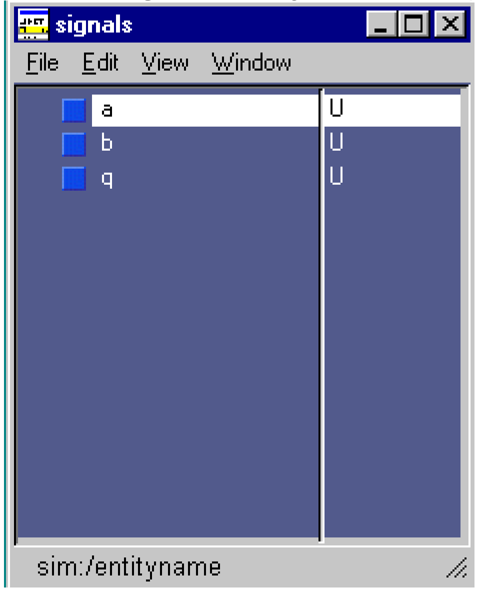


Figure 13 Signals

2. Select **a**.
3. Go to the **Edit** menu and choose **Force**.
4. Force the value **1** for **a**
5. Press **OK**
6. Force the value **0** for **b** the same way.
7. Press **OK**
8. In the wave window single step by pressing (Run) button a few times and watch the expected results.
9. Continue forcing different values during the simulation to see all the possibilities.
10. Close Modelsim window

Exemplar Leonardo Spectrum Synthesis

1. Press the **Synthesis Tool** button, a dialog appears, see **Figure 14.**

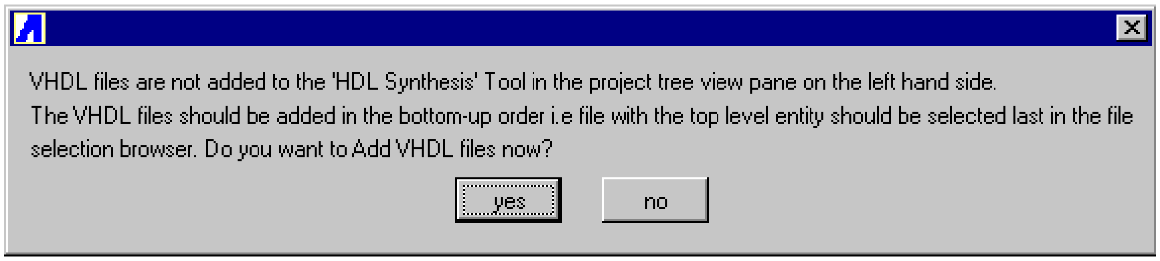
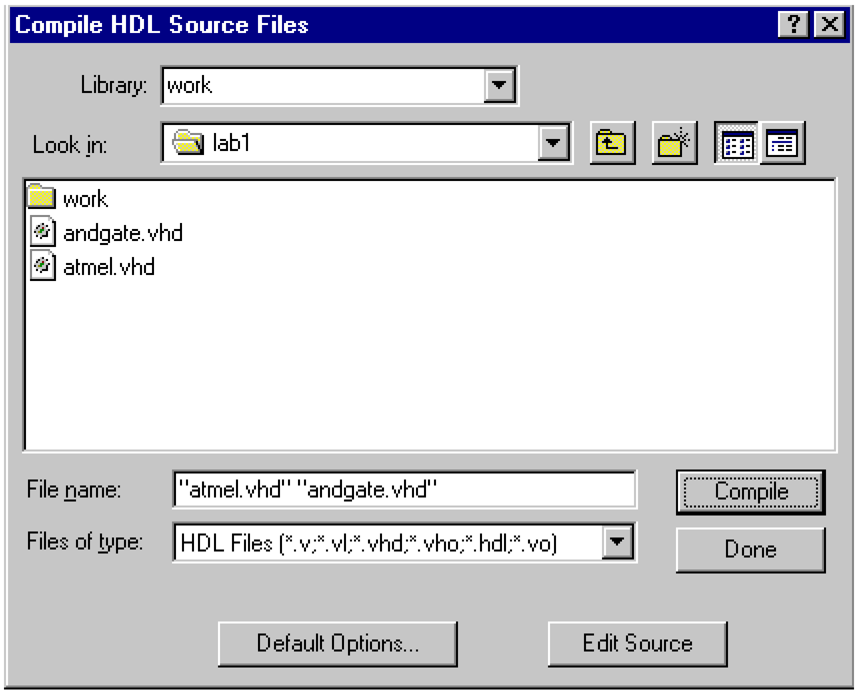

VHDL files are not added to the 'HDL Synthesis' Tool in the project tree view pane on the left hand side.
The VHDL files should be added in the bottom-up order i.e file with the top level entity should be selected last in the file selection browser. Do you want to Add VHDL files now?

Figure 14 VHDL Files

2. Pess **yes**. A browser window appears, see **Figure 15**.



Library: work

Look in: lab1

work
andgate.vhd
atmel.vhd

File name: "atmel.vhd" "andgate.vhd"

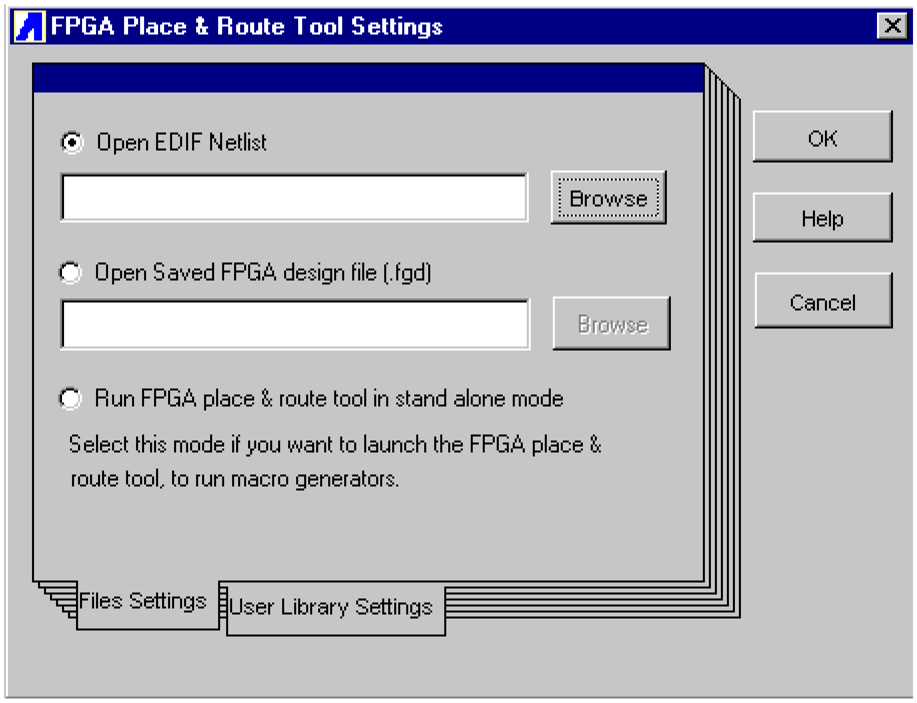
Files of type: HDL Files (*.v;*.vl;*.vhd;*.vho;*.hdl;*.vo)

Figure 15 Browser

3. Select **andgate.vhd** and **atmel.vhd.** Leonardo automatically lists the two selected input files, AT94K as technology and Output file as `c:\training\lab1\andgate.edf`
4. Press the **Run Flow** button.
5. Close Leonardo.

Place and Route IDS (Figaro)

1. Press the Figaro IDS button, a dialog appears, see **Figure 16**.


FPGA Place & Route Tool Settings

[✓ Open EDIF Netlist]

Browse

Open Saved FPGA design file (.fgd)

Browse

Run FPGA place & route tool in stand alone mode

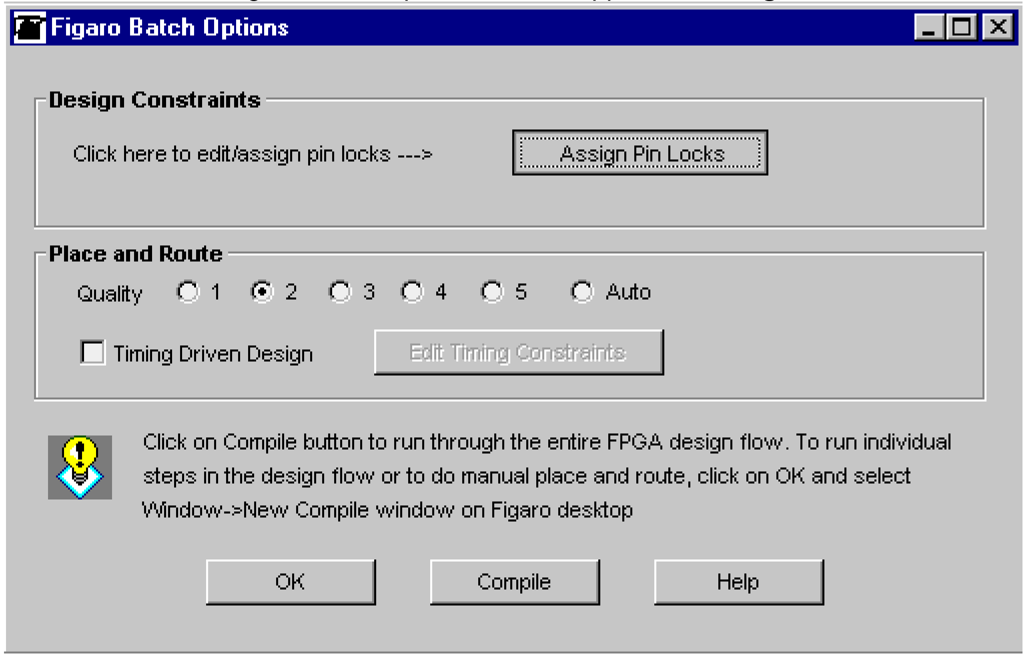
Select this mode if you want to launch the FPGA place & route tool, to run macro generators.

Files Settings | User Library Settings

OK | Help | Cancel

Figure 16 FPGA Place and Route Tool Settings

2. Select **Open EDF Netlist** and press **Browse** to select `c:\training\lab1\andgate.edf` .
3. Press OK, the Figaro Batch Options window appears, see **Figure 17.**


Figaro Batch Options

Design Constraints

Click here to edit/assign pin locks ---> Assign Pin Locks

Place and Route

Quality 1 [✓ 2] 3 4 5 Auto

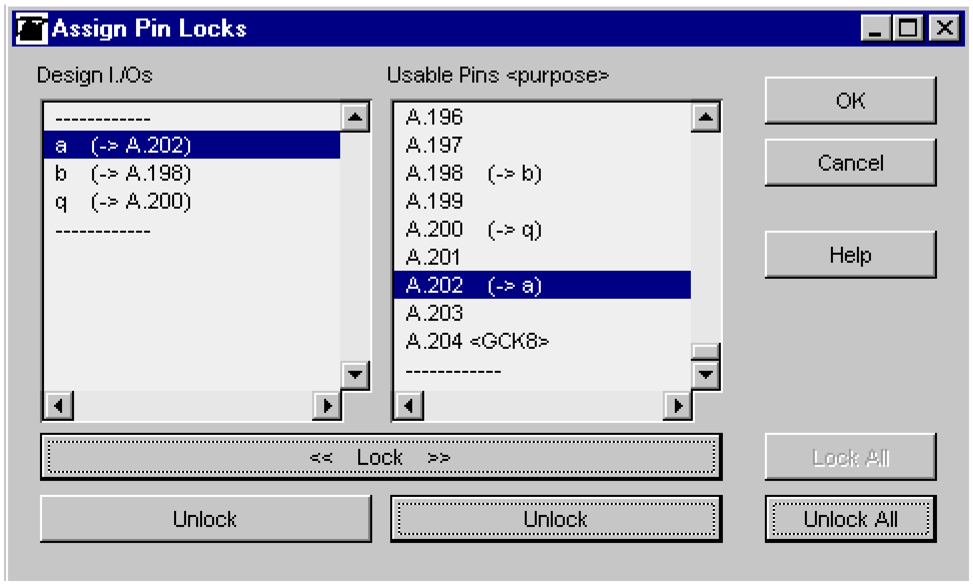
Timing Driven Design (unchecked) Edit Timing Constraints

Click on Compile button to run through the entire FPGA design flow. To run individual steps in the design flow or to do manual place and route, click on OK and select Window->New Compile window on Figaro desktop

OK | Compile | Help

Figure 17 Figaro Batch Options

4. Select Assign Pin Locks, this will open the assign pin locks dialog box, see **Figure 18**.



```
------------
a   (-> A.202)
b   (-> A.198)
q   (-> A.200)
------------
```

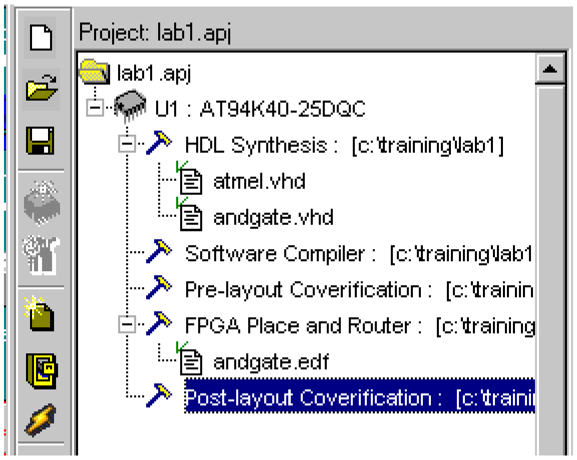
```
A.196
A.197
A.198   (-> b)
A.199
A.200   (-> q)
A.201
A.202   (-> a)
A.203
A.204 <GCK8>
------------
```

Figure 18 Assign Pin Locks

5. Select 'a' under **Design I/O** and 'A.202' under **Usable Pins**
6. Press **Lock**.
7. Follow same procedure for all the I/Os as shown above.
8. Press **OK**
9. Press the **Compile** button. This will partition, place and route the design and export the bitstream.
10. If the routing is successful compile button in Figaro turns **green** or **red** if unsuccessful.
11. Exit Figaro, save the design.

Postlayout Simulation using Modelsim simulator for Lab1

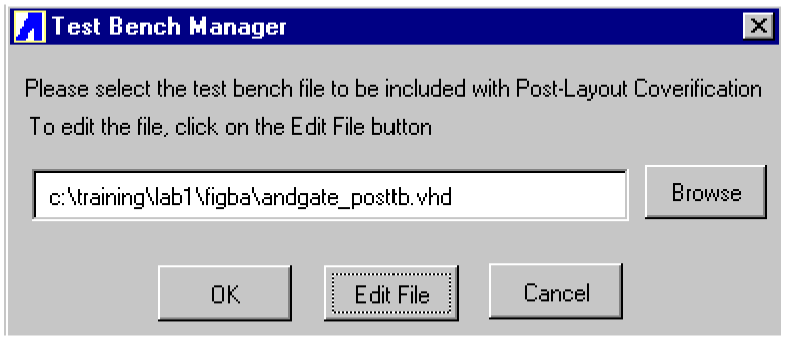
1. Double-click on **Post-layout Coverification** under the Project list to add the back annotated files, see **Figure 19.**



```
lab1.apj
U1 : AT94K40-25DQC
  HDL Synthesis :  [c:\training\lab1]
    atmel.vhd
    andgate.vhd
  Software Compiler :  [c:\training\lab1
  Pre-layout Coverification :  [c:\trainin
  FPGA Place and Router :  [c:\training
    andgate.edf
  Post-layout Coverification :  [c:\traini
```

Figure 19. Project Window

2. The Test Bench Manager window appears, see **Figure 20.**



Please select the test bench file to be included with Post-Layout Coverification
To edit the file, click on the Edit File button

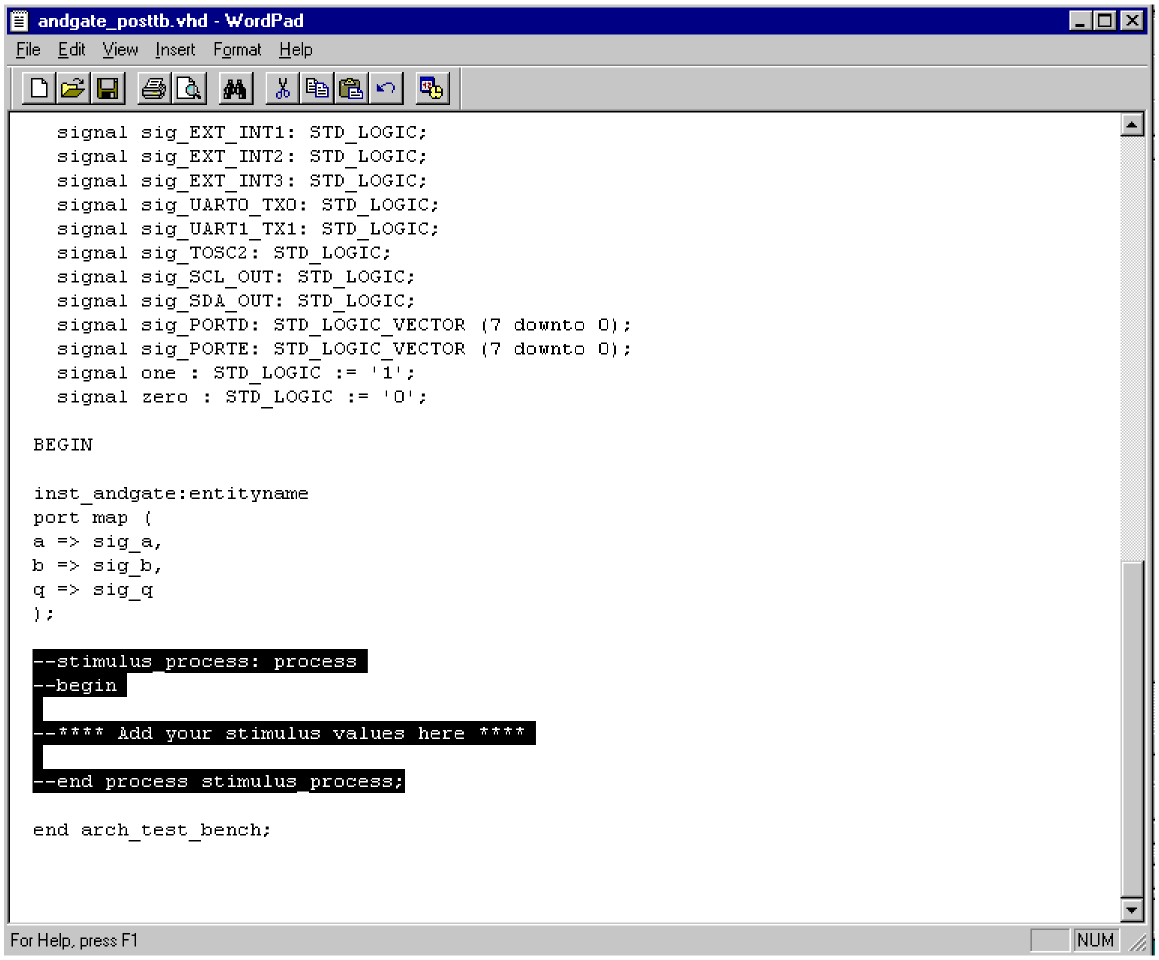
c:\training\lab1\figba\andgate_posttb.vhd

Figure 20 Test Bench Manager

3. Select **Edit File**

Note:
Since we are using only FPGA portion of FPSLIC, it is necessary to delete avr_ram_int component.

4. Highlight the stimulus syntax as shown in **Figure 21.**


andgate_posttb.vhd - WordPad
File Edit View Insert Format Help
For Help, press F1
NUM

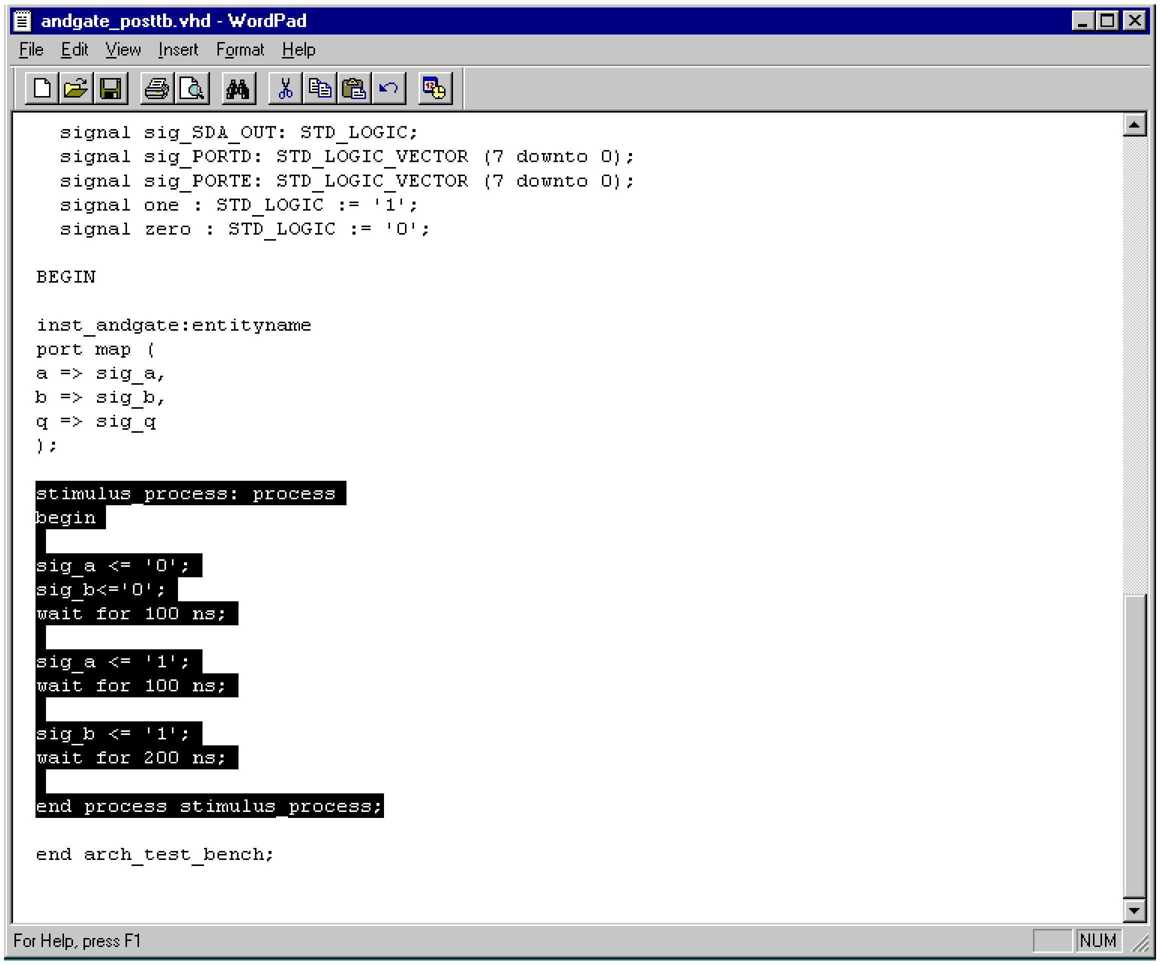
```
  signal sig_EXT_INT1: STD_LOGIC;
  signal sig_EXT_INT2: STD_LOGIC;
  signal sig_EXT_INT3: STD_LOGIC;
  signal sig_UART0_TX0: STD_LOGIC;
  signal sig_UART1_TX1: STD_LOGIC;
  signal sig_TOSC2: STD_LOGIC;
  signal sig_SCL_OUT: STD_LOGIC;
  signal sig_SDA_OUT: STD_LOGIC;
  signal sig_PORTD: STD_LOGIC_VECTOR (7 downto 0);
  signal sig_PORTE: STD_LOGIC_VECTOR (7 downto 0);
  signal one : STD_LOGIC := '1';
  signal zero : STD_LOGIC := '0';

BEGIN

inst_andgate:entityname
port map (
a => sig_a,
b => sig_b,
q => sig_q
);

--stimulus_process: process
--begin

--**** Add your stimulus values here ****

--end process stimulus_process;

end arch_test_bench;
```

Figure 21 Stimulus

5. Copy the following stimulus:

```
stimulus_process: process
begin

sig_a <= '0';
sig_b <= '0';
wait for 100 ns;

sig_a <= '1';
wait for 100 ns;

sig_b <= '1';
wait for 200 ns;

end process stimulus_process;
```

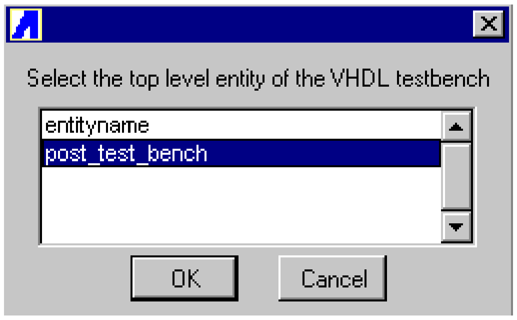
The edited file is shown in **Figure 22.**


andgate_posttb.vhd - WordPad
File Edit View Insert Format Help
For Help, press F1
NUM

```
   signal sig_SDA_OUT: STD_LOGIC;
   signal sig_PORTD: STD_LOGIC_VECTOR (7 downto 0);
   signal sig_PORTE: STD_LOGIC_VECTOR (7 downto 0);
   signal one : STD_LOGIC := '1';
   signal zero : STD_LOGIC := '0';

 BEGIN

 inst_andgate:entityname
 port map (
 a => sig_a,
 b => sig_b,
 q => sig_q
 );

 stimulus_process: process
 begin

 sig_a <= '0';
 sig_b<='0';
 wait for 100 ns;

 sig_a <= '1';
 wait for 100 ns;

 sig_b <= '1';
 wait for 200 ns;

 end process stimulus_process;

 end arch_test_bench;
```

Figure 22 Edited Stimulus

6. Go to the **File** menu and choose **Save.**
7. Press **OK** to close the Test Bench Manager, the *select top level entity of the VHDL test bench* dialog box appears, see **Figure 23**.

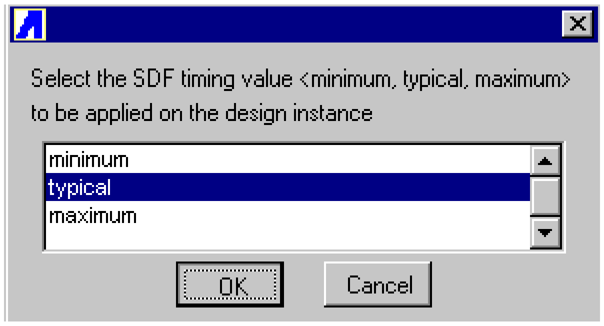

Select the top level entity of the VHDL testbench

entityname

post_test_bench

Figure 23 Top Level Entity

8. Select **post_test_bench** and press **OK**, the SDF timing value dialog appears, see **Figure 24**.


Select the SDF timing value <minimum, typical, maximum> to be applied on the design instance

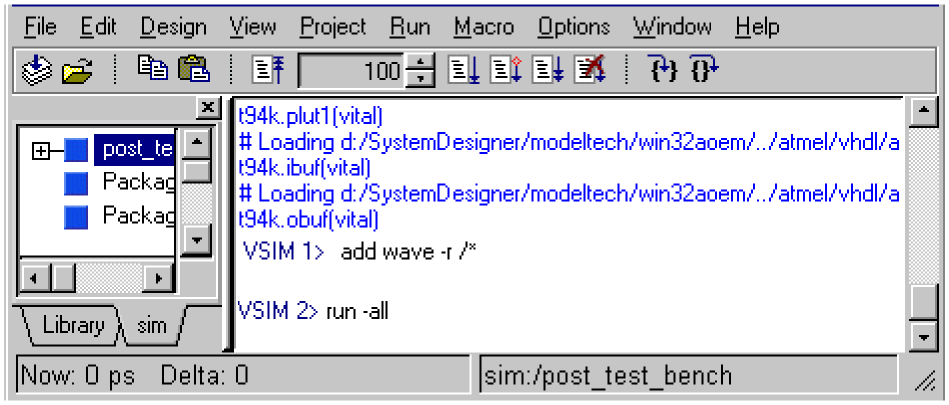
minimum

typical

maximum

Figure 24 SDF Timing Value

9. Select **typical** and press **OK**, ModelSim opens, see **Figure 25**.

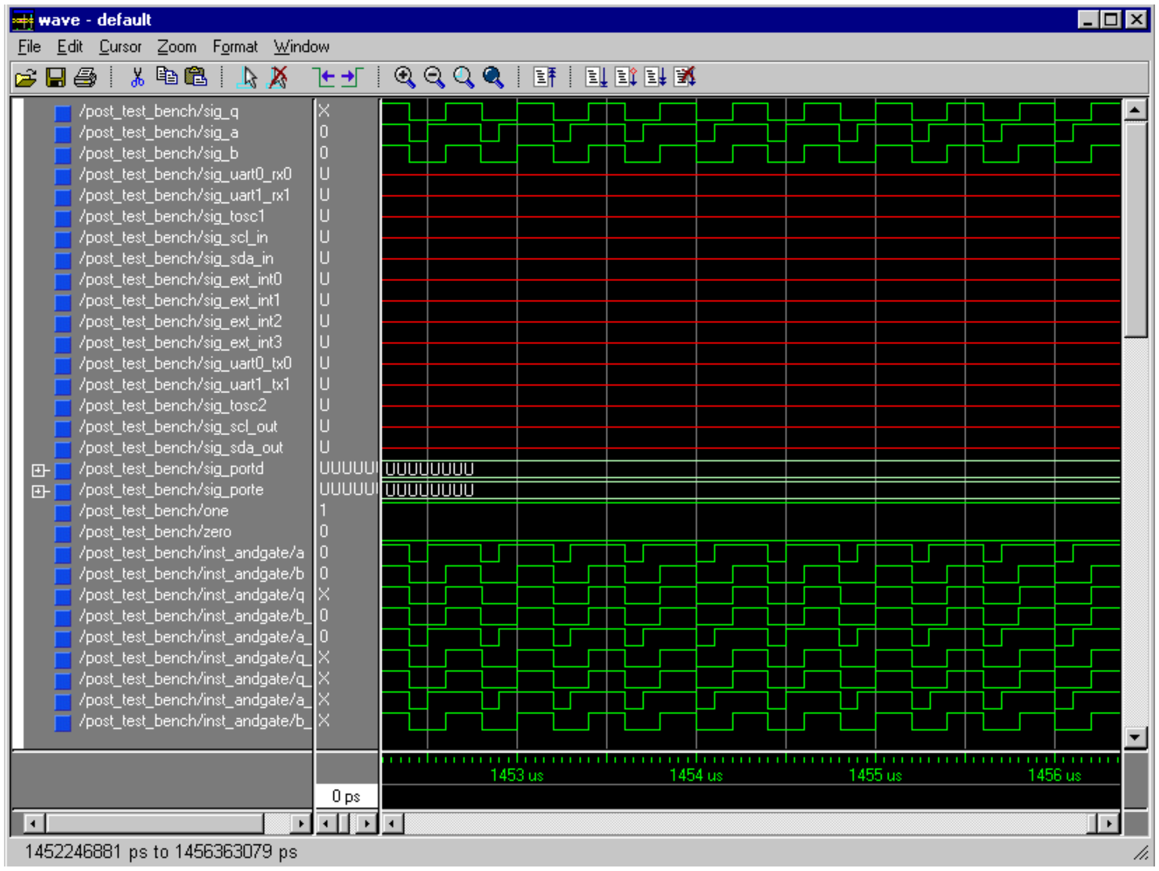


```
t94k.plut1(vital)
# Loading d:/SystemDesigner/modeltech/win32aoem/../atmel/vhdl/a
t94k.ibuf(vital)
# Loading d:/SystemDesigner/modeltech/win32aoem/../atmel/vhdl/a
t94k.obuf(vital)
 VSIM 1> add wave -r /*

VSIM 2> run -all
```

Now: 0 ps   Delta: 0   sim:/post_test_bench

Figure 25 ModelSim

10. Minimize the DOS window
11. Type **add wave –r /***
12. Type **run –all**
13. Press the **break** button after few seconds to see the results, see **Figure 26**.



```
/post_test_bench/sig_q              X
/post_test_bench/sig_a              0
/post_test_bench/sig_b              0
/post_test_bench/sig_uart0_rx0      U
/post_test_bench/sig_uart1_rx1      U
/post_test_bench/sig_tosc1          U
/post_test_bench/sig_scl_in         U
/post_test_bench/sig_sda_in         U
/post_test_bench/sig_ext_int0       U
/post_test_bench/sig_ext_int1       U
/post_test_bench/sig_ext_int2       U
/post_test_bench/sig_ext_int3       U
/post_test_bench/sig_uart0_tx0      U
/post_test_bench/sig_uart1_tx1      U
/post_test_bench/sig_tosc2          U
/post_test_bench/sig_scl_out        U
/post_test_bench/sig_sda_out        U
/post_test_bench/sig_portd          UUUUU  UUUUUUUU
/post_test_bench/sig_porte          UUUUU  UUUUUUUU
/post_test_bench/one                1
/post_test_bench/zero               0
/post_test_bench/inst_andgate/a     0
/post_test_bench/inst_andgate/b     0
/post_test_bench/inst_andgate/q     X
/post_test_bench/inst_andgate/b_    0
/post_test_bench/inst_andgate/a_    0
/post_test_bench/inst_andgate/q_    X
/post_test_bench/inst_andgate/q_    X
/post_test_bench/inst_andgate/a_    X
/post_test_bench/inst_andgate/b_    X
```

1452246881 ps to 1456363079 ps

Figure 26 Wave Results

14. Go to the **File** menu and select **Quit** .

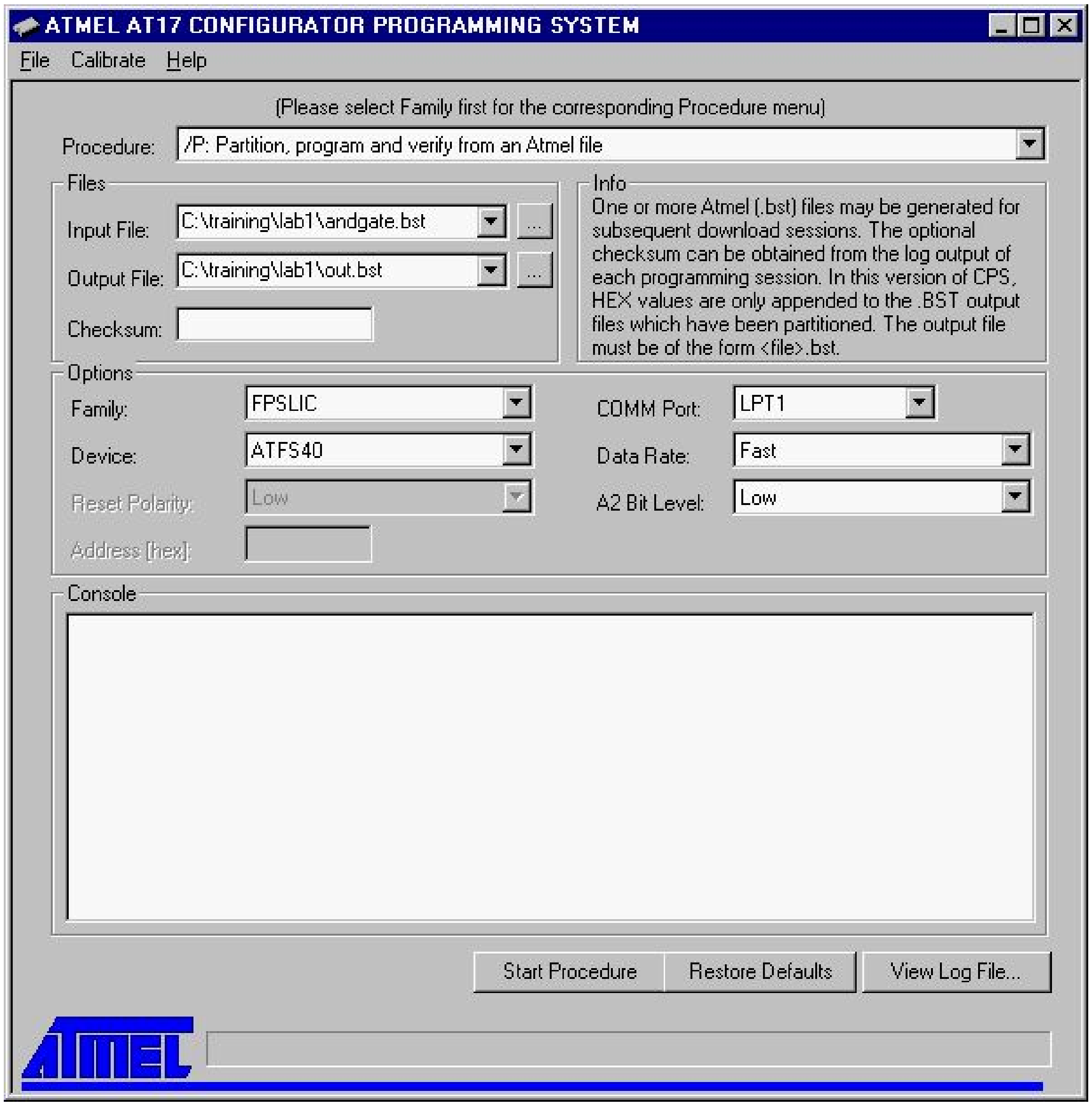
Programming the Device

Hardware Setup

1. Connect the 25-pin parallel cable to the 25-pin Male connector of the ATDH2225 download cable, the 10-pin female header plugs into 10-pin male header **(J1)** on ATDH94STKB board.
2. Connect the power supply from an AC outlet to the 9V DC connector **(P3)** on ATDH94STKB
3. Make sure to set the **jumpers** that are located in between **LEDs** and **Switches** to FPGA side.
4. Adjust **SW10** to **PROG.**
5. Adjust **SW14** to **ON** position.

Software Setup

1. Launch CPS
2. Make sure to set the fields as shown below



(Please select Family first for the corresponding Procedure menu)

Procedure: /P: Partition, program and verify from an Atmel file

Files
Input File: C:\training\lab1\andgate.bst
Output File: C:\training\lab1\out.bst
Checksum:

Info
One or more Atmel (.bst) files may be generated for subsequent download sessions. The optional checksum can be obtained from the log output of each programming session. In this version of CPS, HEX values are only appended to the .BST output files which have been partitioned. The output file must be of the form <file>.bst.

Options
Family: FPSLIC
Device: ATFS40
Reset Polarity: Low
Address [hex]:
COMM Port: LPT1
Data Rate: Fast
A2 Bit Level: Low

Console

Figure 27 CPS

3. Click **Start Procedure,** if everything goes OK you should see “Number of Fatal Errors
4. **Check if AT94K40 configuration is successful:**
   - Press Reset switch **(SW12)** on the right hand board edge.
   - Make sure to adjust **SW10** to **RUN** position.
   - Press **SW1** and **SW2** to check if andgate is working as expected.

LAB 2 – LeonardoSpectrum with Automatic Macro Generators

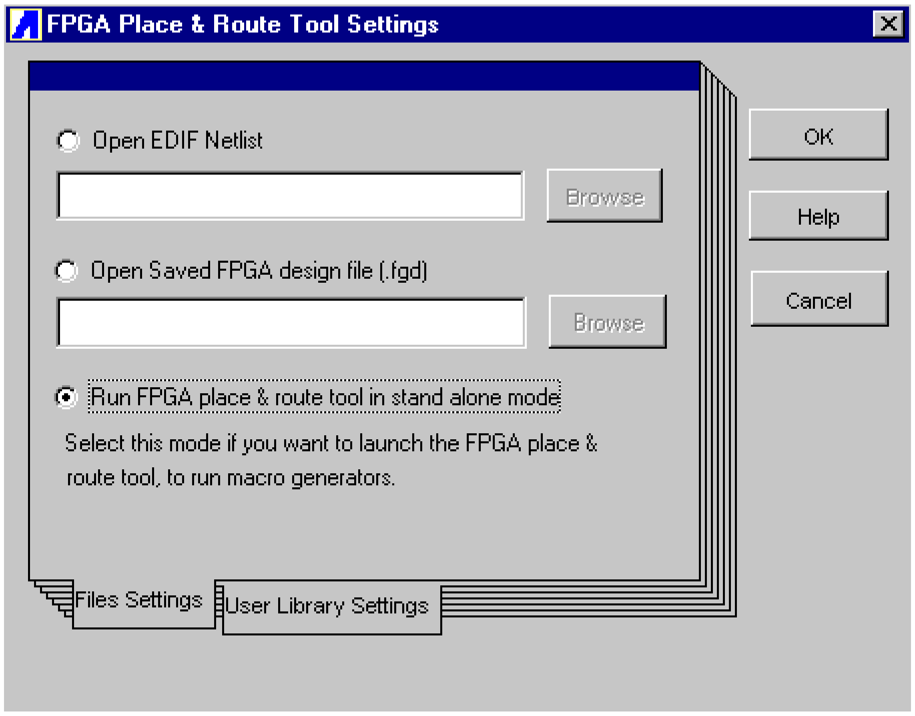
In this lab, the test bench file adder_tb is used for functional simulation.
Copy the file from SystemDesigner/examples/fpga to c:\training\lab1

Creating a Project

- Follow **Section1** in Lab1 to Set up the `c:\training\lab2\lab2.apj` Project in System Designer.

FPGA Place and Route (Figaro IDS) and HDL Planner

1. Press **Figaro IDS**, the FPGA Place and Route Tool Settings dialog box opens, see **Figure 28**.


FPGA Place & Route Tool Settings

( ) Open EDIF Netlist

( ) Open Saved FPGA design file (.fgd)

(●) Run FPGA place & route tool in stand alone mode

Select this mode if you want to launch the FPGA place & route tool, to run macro generators.

Figure 28. FPGA Place and Route Tool Settings

2. Select **Run FPGA place & route tool in stand alone mode**, another FPGA Place and Route Tool Settings dialog box appears.
3. Take the default user library
4. Press **OK.** Figaro opens.
5. Go to the **File** menu and choose **Design Setup.**
6. Press **New Design…**

7. Complete the fields as shown below:

• Design Directory:	**C:\training\lab2**
• Design Name:	**adder**
• Files of Type	**EDIF Netlist (*.edf)**
• Configuration	**AT94K**
• Tools Flow	**Exemplar-MTI**

8. Press OK
9. Press OK

HDL Planner

- Edit the template to match the adder (`Adder.vhd`) design file below:

<u>Adder.vhd</u>

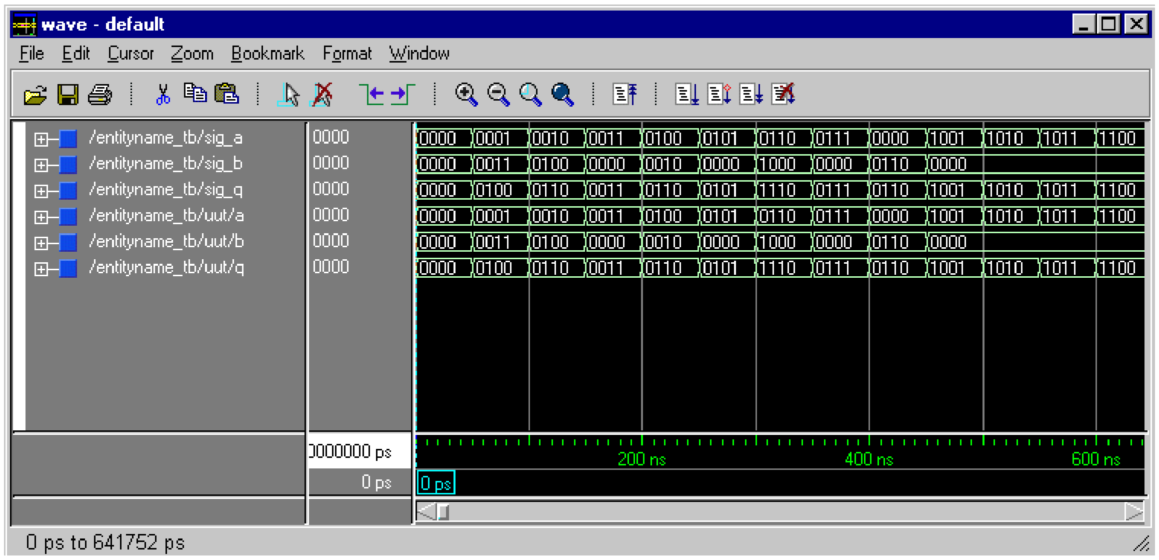
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_signed.all;
--
-- Do not delete  following library and use clauses.
--
library work;
use work.components.all;

ENTITY entityName IS
      PORT (
            A, B        : IN std_logic_vector(3 downto 0);
            Q           : OUT std_logic_vector(3 downto 0)
      );
END entityName;

ARCHITECTURE behaviour of entityName IS
--
--Add SIGNAL definitions here
--
BEGIN
Q <= A +B;
END behaviour;
```

10. Go to the **File** menu and choose **Save As** `adder.vhd` in the directory created above.
11. Close Figaro.

Functional Simulation using test bench file (adder_tb.vhd)

1. Copy `adder_tb` to `c:\training\lab2`
2. Click on the **Simulator** button. ModelSim opens.
3. Go to the **Design** menu and choose **Create a New Library**, take the default
4. Go to the **Design** menu and choose **Compile**, select `compile atmel.vhd, adder.vhd and adder_tb.vhd`
5. Press **Done**
6. Go to the **Design** menu and choose **Load New Design**, select `entityname_tb`
7. Press **Add**
8. Press **Load**
9. Type **add wave –r /***
10. Type **run –all**
11. Click on break button after a few seconds.

12. In the wave window, go to the **Zoom** menu and choose **Zoom Full**. The results are as shown in **Figure 29**.



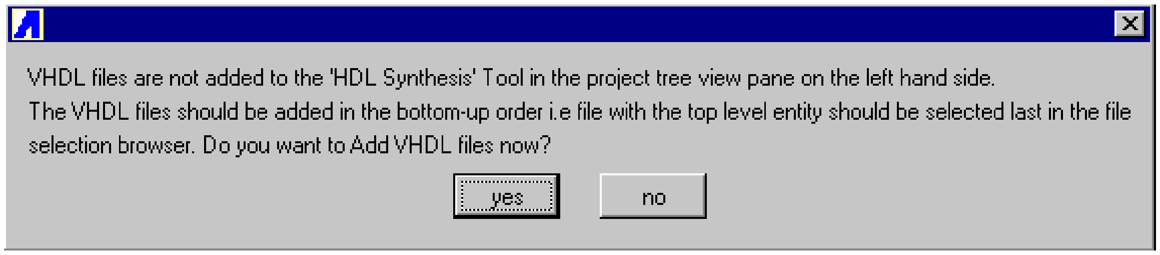
```
/entityname_tb/sig_a  0000  0000 0001 0010 0011 0100 0101 0110 0111 0000 1001 1010 1011 1100
/entityname_tb/sig_b  0000  0000 0011 0100 0000 0010 0000 1000 0000 0110 0000
/entityname_tb/sig_q  0000  0000 0100 0110 0011 0110 0101 1110 0111 0110 1001 1010 1011 1100
/entityname_tb/uut/a  0000  0000 0001 0010 0011 0100 0101 0110 0111 0000 1001 1010 1011 1100
/entityname_tb/uut/b  0000  0000 0011 0100 0000 0010 0000 1000 0000 0110 0000
/entityname_tb/uut/q  0000  0000 0100 0110 0011 0110 0101 1110 0111 0110 1001 1010 1011 1100

0000000 ps        200 ns        400 ns        600 ns
0 ps   0 ps
0 ps to 641752 ps
```

Figure 29 Wave Results

13. In ModelSim, go to the **File** menu and choose **Quit**.

Exemplar Leonardo Spectrum Synthesis

1. Press the **Synthesis Tool** button. A dialog to add the VHDL files appears, see **Figure 30**.



VHDL files are not added to the 'HDL Synthesis' Tool in the project tree view pane on the left hand side.
The VHDL files should be added in the bottom-up order i.e file with the top level entity should be selected last in the file selection browser. Do you want to Add VHDL files now?

Figure 30 Add VHDL Files

2. Pess **yes**. A browser window appears.
3. Select **adder.vhd** and **atmel.vhd**
4. Press OK, Leonardo automatically lists the 2 input files and AT94K as the technology
5. Press **Run Flow**
6. If successful exit Leonardo.

Place and Route using IDS (Figaro)

1. Press the **Figaro IDS** button.
2. Select **Open EDF Netlist** and press **Browse**.
3. Select **adder.edf** andpress **OK**. A dialog to run IDS without the AVR-FPGA constraints appears, see **Figure 31**.

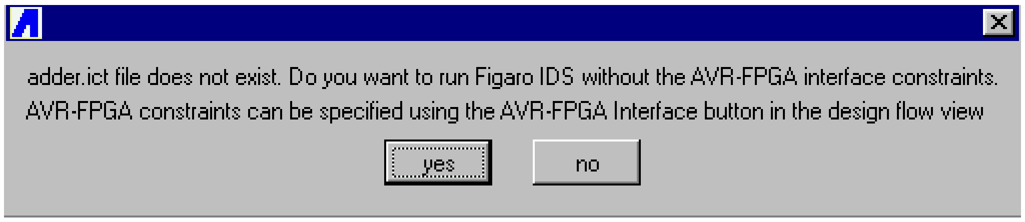

adder.ict file does not exist. Do you want to run Figaro IDS without the AVR-FPGA interface constraints.
AVR-FPGA constraints can be specified using the AVR-FPGA Interface button in the design flow view

Figure 31 AVR-FPGA Constraints

4. Select **yes**, Figaro should run through Open, Map and Parts automatically.
5. Select **Assign Pin Locks**. This will open the pin constraint dialog box.
6. Lock the pins as shown below

Design I/Os	**Usable Pins**	
A0 (SW1)	202	Lock
A1 (SW2)	198	Lock
A2 (SW3)	192	Lock
A3 (SW4)	188	Lock
B0 (SW5)	180	Lock
B1 (SW6)	176	Lock
B2 (SW7)	172	Lock
B3 (SW8)	168	Lock
Q0 (L1)	200	Lock
Q1 (L2)	196	Lock
Q2 (L3)	190	Lock
Q3 (L4)	186	Lock

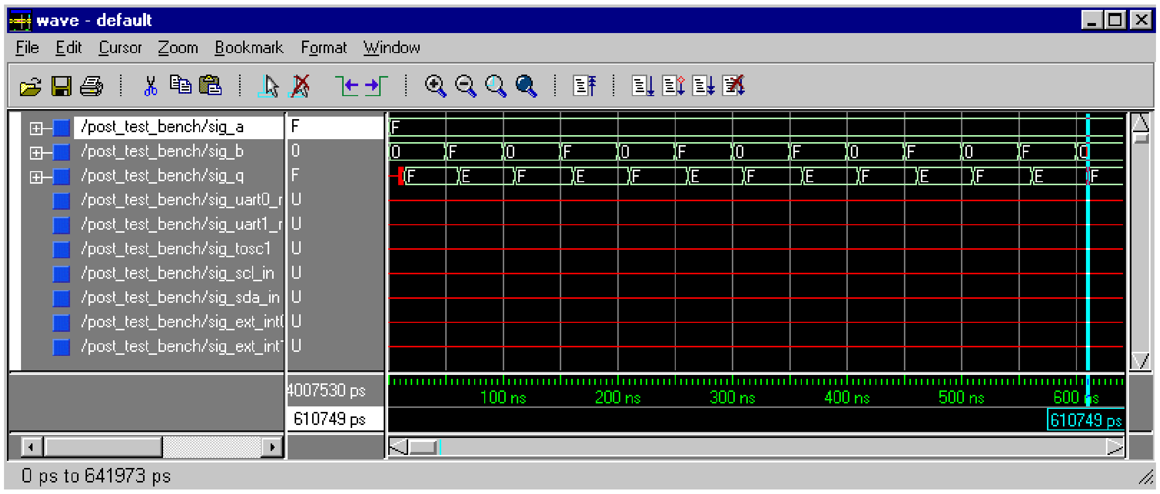
7. Press the **Compile** button to partition, place and route the design and export the bitstream.
8. Go to the **Window** menu and choose **New Compile Window** or double-click on the part in the **Parts Assembler**.
9. Save and close Figaro

Postlayout Simulation

1. Double-click on **Post-layout Coverification** under Project list to add the back annotated files.
2. The Test Bench Manger pops up with the path **c:\training\lab2\figba\adder_posttb.vhd,** press **Edit File**
3. Remove avr_ram_int component and avr_instance: from adder_posttb.vhd (refer lab1)
4. Add the following stimulus to the test bench
```
    sig_a <= “1111”;
    sig_b <= “0000”;
    wait for 50 ns;

    sig_b <= “1111”;
    wait for 50 ns;
```
5. Go to the **File** menu and choose **Save**
6. Close the editor and press **OK**

7. In the next dialog box, select **Post_test_bench and** press **OK**
8. In the next dialog box, select **typical** and press **OK**
9. In the Modelsim window, type **add wave –r /*** and type **run –all**
10. In the Wave window, press the **Break** button to stop simulation.
11. Go to the **Zoom** menu and choose **Zoom Full** to view the results
12. Ctrl select the signals **a_3, a_2, a_1, a_0** in the order specified.
13. Right mouse click and press **Combine**

To change the binary values to hexadecimal or to any other form
1. Select the signal that you wish to change the radix
2. Right click and select **Radix** to change the results from binary to hexadecimal or any other form. The final results are as shown in **Figure 32**.



```
/post_test_bench/sig_a         F
/post_test_bench/sig_b         0
/post_test_bench/sig_q         F
/post_test_bench/sig_uart0_r   U
/post_test_bench/sig_uart1_r   U
/post_test_bench/sig_tosc1     U
/post_test_bench/sig_scl_in    U
/post_test_bench/sig_sda_in    U
/post_test_bench/sig_ext_int0  U
/post_test_bench/sig_ext_int1  U
4007530 ps
610749 ps
0 ps to 641973 ps
```

Figure 32 Wave Results

3. Go to the **File** menu and choose **Quit**

Programming the Device using adder.bst

Refer to lab 1.

LAB 3 : Synthesis with Macros

Creating a Project

1. Create a new named **C:\training\lab3\lab3.apj**

FPGA Place and Route

1. Press the IDS Figaro button.
2. Select **Run FPGA place & route tool in stand alone mode**, another FPGA Place and Route Tool Settings dialog box appears.
3. Take the default user library
4. Press **OK.** Figaro opens.
5. Press **File** → **Design Setup.**
6. Press **New Design… c**omplete the fields as shown below:

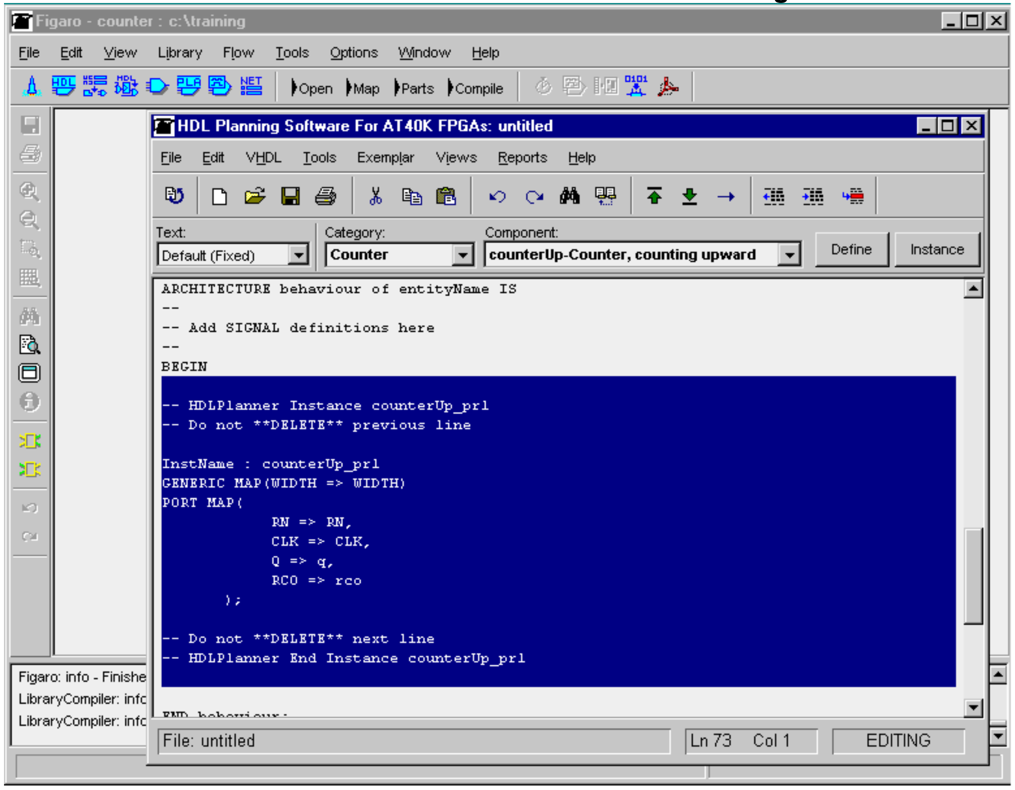
• Design Directory:	**C:\training\lab3**
• Design Name:	**counter**
• Files of Type	**EDIF Netlist (*.edf)**
• Configuration	**AT94K**
• Tools Flow	**Exemplar-MTI**

7. Press OK
8. Press OK

HDL Planner

Start HDLPlanner.

1. From the **Category_** pull-down list, select Counter
2. From the **Component** pull-down list, select **CounterUp – Counter, counting Upward**
3. Press the **Define**, take the default.
4. Scroll down to the bottom of the file and select as shown in **Figure 33**..



Figaro - counter : c:\training
File Edit View Library Flow Tools Options Window Help
Open Map Parts Compile
HDL Planning Software For AT40K FPGAs: untitled
File Edit VHDL Tools Exemplar Views Reports Help
Text: Default (Fixed)
Category: Counter
Component: counterUp-Counter, counting upward
Define
Instance
Figaro: info - Finishe
LibraryCompiler: infc
LibraryCompiler: infc
File: untitled
Ln 73 Col 1
EDITING

```
ARCHITECTURE behaviour of entityName IS
--
-- Add SIGNAL definitions here
--
BEGIN

-- HDLPlanner Instance counterUp_prl
-- Do not **DELETE** previous line

InstName : counterUp_prl
GENERIC MAP(WIDTH => WIDTH)
PORT MAP(
                RN => RN,
                CLK => CLK,
                Q => q,
                RCO => rco
        );

-- Do not **DELETE** next line
-- HDLPlanner End Instance counterUp_prl
```

Figure 33 Signal Definitions

5. Then press the **Instance** button to instantiate the code.

Note - **Define** will always place the component definition in the correct place.
Instance will instantiate the code wherever the cursor is.

6. Edit the template to match the counter.vhd design as shown in next two pages (need to only edit the second page). Sections shown in italics are supplied by HDL Planner, sections in bold should be hand edited.

Counter.vhd

```
-- HDLPlanner Start counterUp_prl
-- Do not **Delete** previous line

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

ENTITY counterUp_prl IS
        generic (WIDTH   : integer := 4);
        port(
              RN       : IN std_logic;
              CLK     : IN std_logic;
              Q         : OUT std_logic_vector (WIDTH-1 downto 0);
              RCO       : OUT std_logic
          );
END counterUp_prl ;

ARCHITECTURE behv OF counterUp_prl IS

constant clockEdge : std_logic := '1';
constant setOrResetLevel : std_logic := '0';
constant setOrReset: integer := 0;


SIGNAL Qtmp : std_logic_vector(WIDTH-1 downto 0);

begin
    process(CLK,RN)
    begin
         if(RN = setOrResetLevel) then
                Qtmp <= CONV_STD_LOGIC_VECTOR(setOrReset,WIDTH);
         elsif (CLK = clockEdge and CLK'event) then
                Qtmp <= Qtmp + '1';
         end if;

         if (Qtmp = CONV_STD_LOGIC_VECTOR(-1,WIDTH)) then
                RCO <= '1';
         else
                RCO <= '0';
         end if;

     end process;

         Q <= Qtmp;

end behv;
-- Do not **Delete** next line
-- HDLPlanner End counterUp_prl


LIBRARY ieee ;
USE ieee.std_logic_1164.all;
--
-- Do not delete  following library and use clauses.
--
library work;
use work.components.all;

ENTITY entityName IS
      PORT (
                  RESET : IN std_logic;
                  CLOCK : IN std_logic;
                  COUNT : OUT std_logic_vector ( 3 downto 0);
                  RCOUT : OUT std_logic
      );
END entityName;

ARCHITECTURE behaviour of entityName IS
--
-- Add SIGNAL definitions here
```

```
--
BEGIN

-- HDLPlanner Instance counterUp_prl
-- Do not **DELETE** previous line

InstName : counterUp_prl
GENERIC MAP(WIDTH => 4)
PORT MAP(
            RN => RESET,
            CLK => CLOCK,
            Q => COUNT,
            RCO => RCOUT
            );

-- Do not **DELETE** next line
-- HDLPlanner End Instance counterUp_prl


END behaviour;
```

These are the parameters for the Macro Generator. You can use the same definition for any counter, and simply instantiate different sizes by modifying the values above and the net names.

7. Go to the **Tools** menu and choose **Macro Generators**. This will create any components you have instantiated.
8. Go to the **File** menu and choose **Save As** `counter.vhd`
9. Close Figaro

Exemplar Leonardo Spectrum Synthesis

1. Press on **Synthesis Tool button**(This will open Leonardo Spectrum)
2. Proceed as in earlier examples

Place and Route using IDS (Figaro)

1. Select **Figaro IDS…**
2. Enable **Open EDF Netlist**
3. Press **Browse**, select **counter.edf** in the next dialog box
4. Press **OK**
5. Select **yes**, when the next dialog pops up, Figaro should run through Open, Map and Parts automatically.
6. Select **Assign Pin Locks**. This will open the pin constraint dialog box.
7. Press **OK**
8. Press the **Compile** button. This will partition, place and route the design and export the bitstream.
9. Go to the **Window** menu and choose **New Compile Window** or double-click on the part in the Parts Assembler
10. Save and close Figaro

Postlayout Simulation

1. Double click on **Post-layout Coverification** under Project list to add the back annotated files.
2. Test Bench Manger pops up with the path **c:\training\lab2\figba\counter_posttb.vhd**
3. Press **Edit File**
4. Remove avr_ram_int component & avr_instance: from counter_posttb.vhd and add stimulus as shown at the end of the file. Edited portion is in bold:

```
-- Post-Layout Test Bench File
------------------------------------
-- Design : counter
-- Program : Figaro
-- Version : Atmel 7.2 (patch level 1 applied)
-- Vendor : Atmel
-- Created : May 9, 2001 at : 1:42:41 am

library IEEE;
use IEEE.STD_LOGIC_1164.all;

use IEEE.VITAL_timing.all;

library AT94K;
--library user94k;
use AT94K.VCOMPONENTS.all;

entity post_test_bench is
end post_test_bench;

architecture arch_test_bench of post_test_bench is

component entityName
    port (
      RESET : in STD_LOGIC := 'X';
      CLOCK : in STD_LOGIC := 'X';
      COUNT : out STD_LOGIC_VECTOR(3 downto 0);
      RCOUT : out STD_LOGIC
    );
end component;

  signal sig_RCOUT: STD_LOGIC;
  signal sig_COUNT: STD_LOGIC_VECTOR(3 downto 0);
  signal sig_RESET: STD_LOGIC;
  signal sig_CLOCK: STD_LOGIC;
  signal sig_UART0_RX0: STD_LOGIC;
  signal sig_UART1_RX1: STD_LOGIC;
  signal sig_TOSC1: STD_LOGIC;
  signal sig_SCL_IN: STD_LOGIC;
  signal sig_SDA_IN: STD_LOGIC;
  signal sig_EXT_INT0: STD_LOGIC;
  signal sig_EXT_INT1: STD_LOGIC;
  signal sig_EXT_INT2: STD_LOGIC;
  signal sig_EXT_INT3: STD_LOGIC;
  signal sig_UART0_TX0: STD_LOGIC;
  signal sig_UART1_TX1: STD_LOGIC;
  signal sig_TOSC2: STD_LOGIC;
  signal sig_SCL_OUT: STD_LOGIC;
  signal sig_SDA_OUT: STD_LOGIC;
  signal sig_PORTD: STD_LOGIC_VECTOR (7 downto 0);
  signal sig_PORTE: STD_LOGIC_VECTOR (7 downto 0);
  signal one : STD_LOGIC := '1';
  signal zero : STD_LOGIC := '0';

shared variable ENDSIM:boolean := false;
constant CLK_PERIOD: TIME := 200 ns;

BEGIN

inst_counter:entityName
```

```
port map (
RESET => sig_RESET,
CLOCK => sig_CLOCK,
COUNT => sig_COUNT,
RCOUT => sig_RCOUT
);

CLK_GEN : process
begin
  if ENDSIM = false then
    sig_clock <= '1';
      wait for CLK_PERIOD/2;
    sig_clock <= '0';
      wait for CLK_PERIOD/2;
    else
      wait;
    end if;
  end process;

stimulus_process: process
begin

      sig_reset <= '0';
        wait for 50 ns;
      sig_reset <= '1';
        wait for 50 us;

end process stimulus_process;

end arch_test_bench;
```

5. Go to the **File** menu and choose **Save**
6. Press **OK** in the Test Bench Manager dialog box
7. In the next dialog box, select **Post_test_bench** and press **OK**
8. In the next dialog box, select **typical** and press **OK**
9. In the ModelSim window, Type **add wave –r /*** and type **run –all**
10. In the Wave window, press the **Break** button in the wave window to stop simulation.
11. Go to the **Zoom** menu and choose **Zoom Full** to view the results

Programming the Device

Download counter.bst to AT94K device as shown in previous labs.

LAB 4 : LeonardoSpectrum with Black Box Macro Generators

Creating a New Design

- Create a new Project named **C:\training\lab4\lab4.apj**

FPGA Place and Route

1. Select **Run FPGA place & route tool in stand alone mode**, another FPGA Place and Route Tool Settings dialog box appears.
2. Take the default user library
3. Press **OK.** Figaro opens.
4. Go to the **File** menu and choose **Design Setup.**
5. Press **New Design…** and complete the fields as shown below:

• Design Directory:	**C:\training\lab4**
• Design Name:	ramdesign
• Files of Type	**EDIF Netlist (*.edf)**
• Configuration	**AT94K**
• Tools Flow	**Exemplar-MTI**

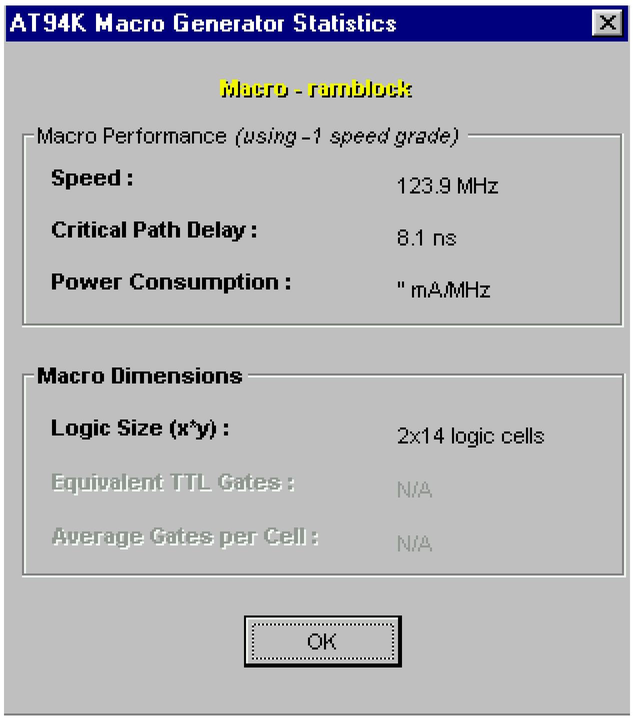
6. Press OK

Macro Generator

1. Start Macro Generator.
2. Fill in the following fields:

• Category Tab	Memory
• Macro Tab	RAM - Dual Port
• Address Width	6
• Width	16
• External Decoding	Clustered
• RAM Type	Synchronous
• Macro Name	ramblock

- Press the **Generate** button. A statistics dialog appears if the generator ran successfully, see **Figure 34**.


AT94K Macro Generator Statistics

Macro - ramblock

Macro Performance *(using -1 speed grade)*

Speed :	123.9 MHz
Critical Path Delay :	8.1 ns
Power Consumption :	" mA/MHz

Macro Dimensions

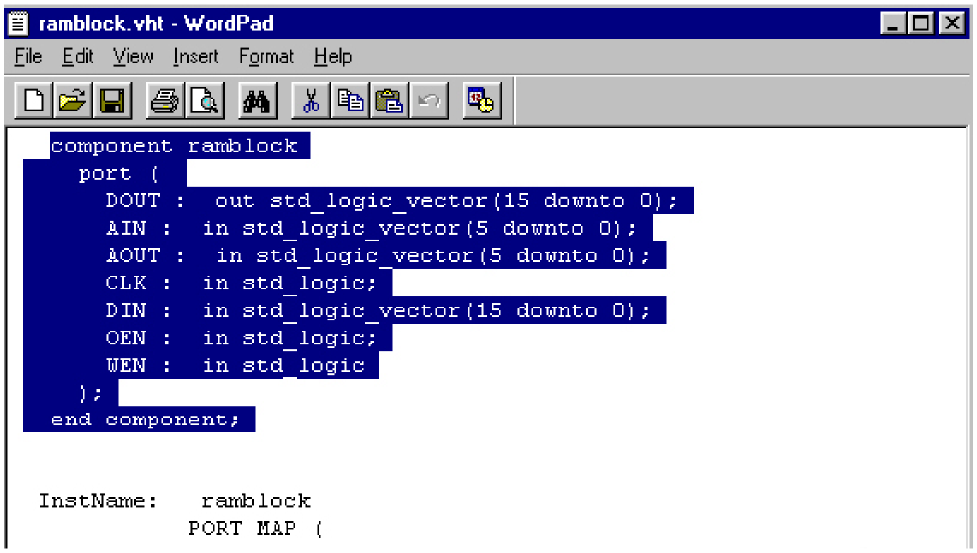
Logic Size (x*y) :	2x14 logic cells
Equivalent TTL Gates :	N/A
Average Gates per Cell :	N/A

Figure 34 Statistics

3. Press **OK** to close the statistics dialog box.
4. Press **Cancel** to close the 'Macro Generator dialog box'.

HDL Planner

1. Edit the template to match the Ramdesign.vhd file on the next page. You can copy the **component** definition from the file **ramblock.vht** under **c:/training/lab4/user94k/ramblock**, see **Figure 35**.



```
  component ramblock
    port (
      DOUT :  out std_logic_vector(15 downto 0);
      AIN :  in std_logic_vector(5 downto 0);
      AOUT :  in std_logic_vector(5 downto 0);
      CLK :  in std_logic;
      DIN :  in std_logic_vector(15 downto 0);
      OEN :  in std_logic;
      WEN :  in std_logic
    );
  end component;


InstName:   ramblock
           PORT MAP (
```

Figure 35 Component

2. **Paste** the component definition under ‘ARCHITECTURE’ as shown in the Ramdesign.vhd file You can do the same for Port Map as well, this time you paste the contents under BEGIN. Italicized contents shows the copied contents from ramdesign.vht file and bold content shows the edited part of it

Ramdesign.vhd

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all;
--
-- Do not delete  following library and use clauses.
--
library work;
use work.components.all;

ENTITY entityName IS
      PORT (            dataout     :  out std_logic_vector(15 downto
0);
                  addressin   :  in std_logic_vector(5 downto 0);
                  addressout  :  in std_logic_vector(5 downto 0);
                        clock             :  in std_logic;
                        datain            :  in std_logic_vector(15
downto 0);
                        read        :  in std_logic;
                        write             :  in std_logic
      );
END entityName;

ARCHITECTURE behaviour of entityName IS
SIGNAL dmmy_din   : STD_LOGIC_VECTOR(7 downto 0) := (others => 'Z' );
SIGNAL dummy      : STD_LOGIC := 'Z';
--
-- Add SIGNAL definitions here
--
component ramblock
    port (
      DOUT :  out std_logic_vector(15 downto 0);
      AIN :  in std_logic_vector(5 downto 0);
      AOUT :  in std_logic_vector(5 downto 0);
      CLK :  in std_logic;
      DIN :  in std_logic_vector(15 downto 0);
      OEN :  in std_logic;
      WEN :  in std_logic
    );
  end component;

BEGIN
U1 : ramblock
      PORT MAP (
                        DOUT => dataout,
                        AIN => addressin,
                        AOUT => addressout,
                        CLK => clock,
                        DIN => datain,
                        OEN => read,
                        WEN => write
      );
END behaviour;
```

3. When complete **Save As** Ramdesign.vhd. Make sure it is in the design directory.

4. Close HDL Planner and Figaro

Exemplar Logic-LeonardoSpectrum Synthesis

- Press the **Synthesis Tool** button.
- Press **yes** to add the VHDL files.
- Select **ramdesign.vhd** and **atmel.vhd**
- Press **OK**, Leonardo automatically lists the 2 input files and AT94K as the technology
- Press **Run Flow**
- Close Leonardo, now need to save the project

Place and Route using IDS (Figaro)

1. Select **Figaro IDS…**
2. Enable **Open EDF Netlist**
3. Press **Browse**, select **ramdesign.edf** in the next dialog box
4. Press **OK**
5. Select **yes**, when the next dialog pops up, Figaro automatically runs Open, Map and Parts.
6. Press the **Compile** button. This will partition, place and route the design and export the bitstream.
7. Go to the **Window** menu and choose **New Compile Window** or double-click on the part in the Parts Assembler
8. Save and close Figaro

Postlayout Simulation

1. Double click on **Post-layout Coverification** under Project list to add the back annotated files.
2. Test Bench Manger pops up with the path **c:\training\lab2\figba\ramdesign_posttb.vhd**
3. Press **Edit File**
4. Remove avr_ram_int component & avr_instance: from ramdesign_posttb.vhd and add the stimulus as shown at the end of the file. Edited portion is in bold:

```
signal sig_PORTE: STD_LOGIC_VECTOR (7 downto 0);
signal one : STD_LOGIC := '1';
signal zero : STD_LOGIC := '0';

shared variable ENDSIM : boolean := false;
constant CLK_PERIOD : TIME := 200 ns;

BEGIN

inst_ramdesign:entityName
port map (
ADDRESSIN => sig_ADDRESSIN,
ADDRESSOUT => sig_ADDRESSOUT,
clock => sig_clock,
DATAIN => sig_DATAIN,
read => sig_read,
write => sig_write,
DATAOUT => sig_DATAOUT
);
CLK_GEN : process
begin
  if ENDSIM = false then
    sig_clock <= '1';
```

```
      wait for CLK_PERIOD/2;
    sig_clock <= '0';
      wait for CLK_PERIOD/2;
    else
      wait;
  end if;
end process;

stimulus_process: process
begin

sig_read <= '1';
sig_write <= '0';

sig_addressin <= "100000";
sig_datain <= "1110000000000000";

  wait for 500 ns;
sig_read <= '0';
sig_addressout <= "100000";
  wait for 500 ns;
end process stimulus_process;

end arch_test_bench;
```

5. Go to the **File** menu and choose **Save**
6. Press **OK** in the Test Bench Manager dialog box
7. In the next dialog box, select **Post_test_bench** and press **OK**
8. In the next dialog box, select **typical** and press **OK**
9. In the ModelSim window, Type **add wave –r /*** and type **run 500us**
10. In the Wave window, press the **Break** button in the wave window to stop simulation.
11. Go to the **Zoom** menu and choose **Zoom Full** to view the results