QUICK START TUTORIAL

Mentor Graphics – Verilog Design Flow

ProChip Designer v5.0

Requirements

Software:

ProChip Designer v5.0

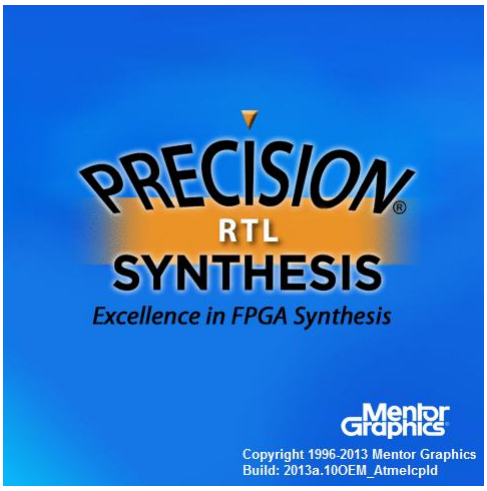
Precision Synthesis

ModelSim (optional)

ATMISP

Hardware:

ATF15xx-DK3-U Kit



Contents

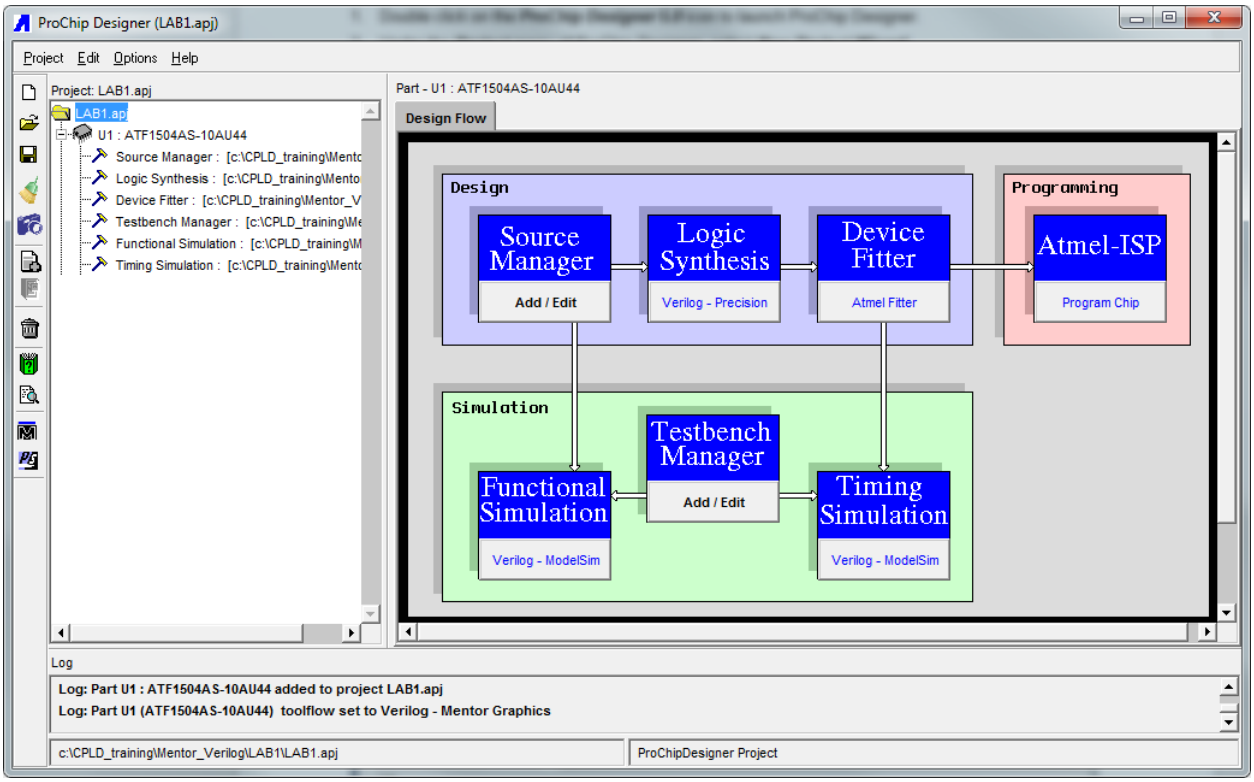
Tutorial 1: 2-Input AND Gate Design

In this tutorial, you will be guided through a complete Verilog design flow using ProChip Designer and Mentor Graphics Precision Synthesis.

Create a New Project in ProChip Designer

1. Double click on the ***ProChip Designer 5.0*** icon to launch ProChip Designer.
2. Under the ***Project*** menu of ProChip Designer, select ***New Project Wizard…***
3. In the **New Project Wizard – Step 1 of 6** window, click on the ***Next >*** button.
4. In **New Project Wizard – Step 2 of 6** window, click on the ***Browse*** button and create a new design directory "**C:\CPLD_training\Mentor_Verilog\LAB1"** and specify "**LAB1.apj**" as the project name. Then click on the ***Next >*** button.
5. In the **Part Number** dialog box of the **New Project Wizard – Step 3 of 6** window, select ***ATF1504AS-10AU44*** (or any ATF15xx 44-TQFP that is available to you) as the target device type. Then click on the ***Next >*** button.
6. In the **Tool flow** dialog box of the **New Project Wizard – Step 4 of 6** window, select ***Verilog – Mentor Graphics***. Then click on the ***Next >*** button.
7. In the **New Project Wizard – Step 5 of 6** window, click on the ***Next >*** button.
8. In the **New Project Wizard – Step 6 of 6** window, click on the ***Finish*** button to close the **New Project Wizard**.

Upon clicking on the **Finish** button, the **Project** (left), **Design Flow** (right) and **Log** (bottom) windows will appear as shown below.

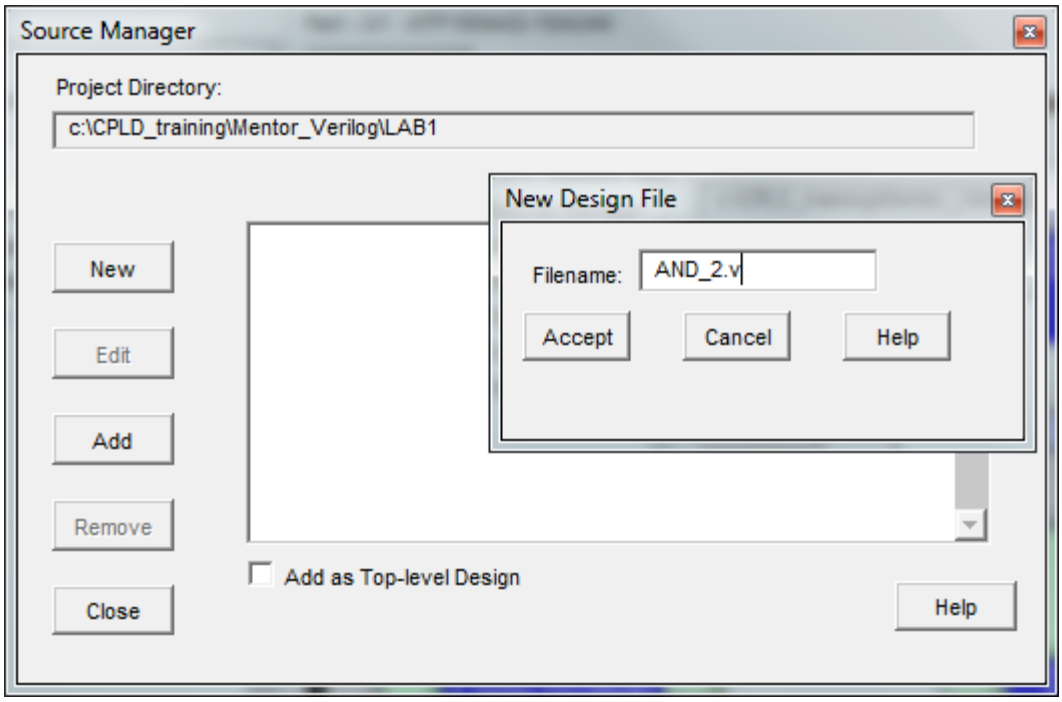


The Design Flow view shows the steps that are required to implement a design in an ATF15xx CPLD using ProChip Designer. The arrows on the diagram show dependencies for the different steps. For example, in order to run Timing Simulation, you must first complete the Source Manager, Logic Synthesis, and then Device Fitter steps to generate the necessary back-annotated simulation files and also setup the testbench file in the Testbench Manager.

Create a New Verilog Design Using HDL Planner

1. Click on the ***Add / Edit*** button under **Source Manager** of the **Design Flow** tab window.

2. Click on the ***New*** button in the **Source Manager** window to open the **New Design File** dialog window.



Note:
When creating or adding design files in this window, the top-level design file must be placed at the top of the file list. Therefore, it must be created/added in this window first before other sub-modules are created/added, or the **Add as Top-level Design** box must be checked before creating/adding the top-level design file.

3. Enter the Verilog design filename "**AND_2.v**" into the **New Design File** dialog window and then click on the ***Accept*** button. The **New module wizard** window will open.

4. In the **New module wizard** window, enter "**AND_2**" into the **Entity name** box.

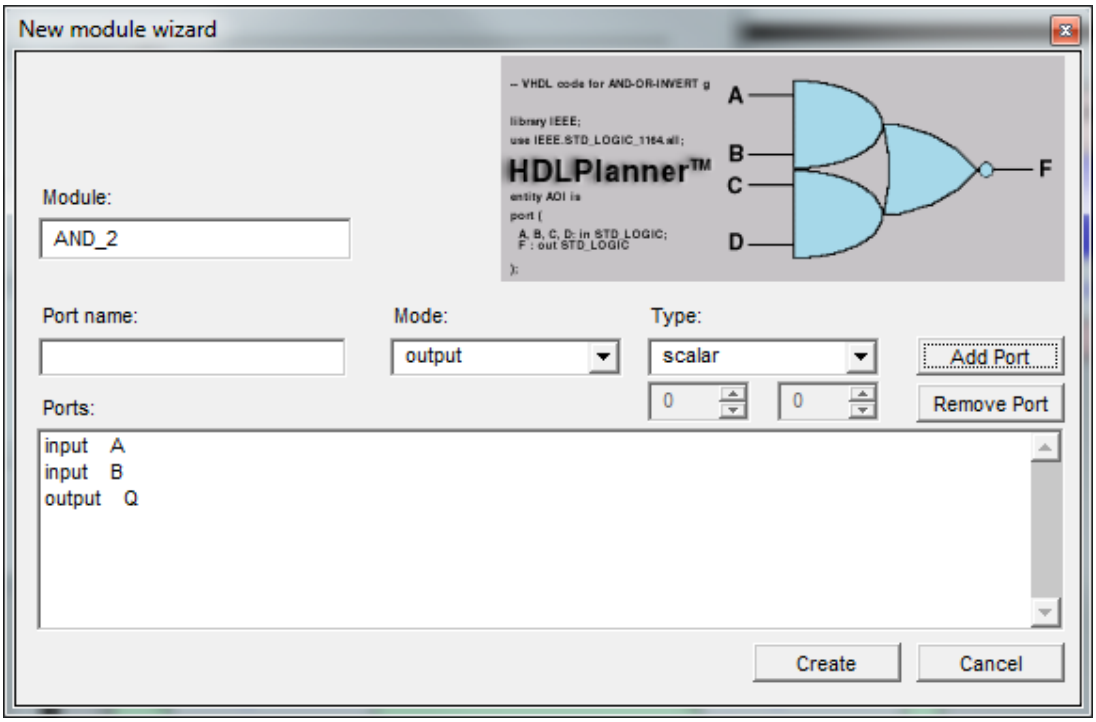
   Note:
   "AND2" cannot be used as the Entity name since "AND2" is the name of a component in the ATF15xx synthesis library.

5. Enter "**A**" into the **Port name** box, select ***Input*** under **Mode** and ***scalar*** under **Type**, and then click on the ***Add Port*** button.

6. Enter "**B**" into the **Port name** box, select ***Input*** under **Mode** and ***scalar*** under **Type**, and then click on the ***Add Port*** button.

7. Enter "**Q**" into the **Port name** box, select ***output*** under **Mode** and ***scalar*** under **Type,** and then click on the ***Add Port*** button.

Now, the **Ports** dialog box will show the input and output ports as shown below.

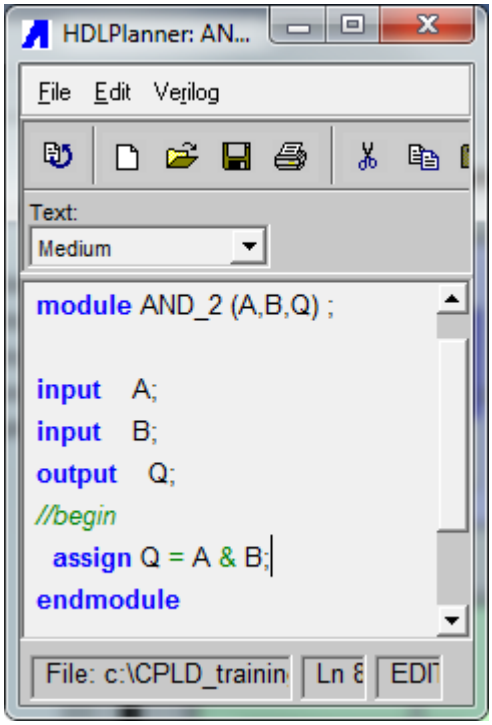


8. Click on the ***Create*** button to create the Verilog design file with the Entity, Input Ports and Output Ports specified in the **New module wizard** and the Verilog design file will open in HDL Planner.

   Note:
   If you do not wish to use the **New module wizard**, you can just click on the ***Cancel*** button in the **New module wizard** and then create the design code in HDL Planner directly.

9. Add the following logic equation to the Verilog design file before "**endmodule**":

```
assign Q = A & B;
```



10. Go to the ***File*** menu in HDL Planer and then select ***Save*** to save your design to the current project directory.

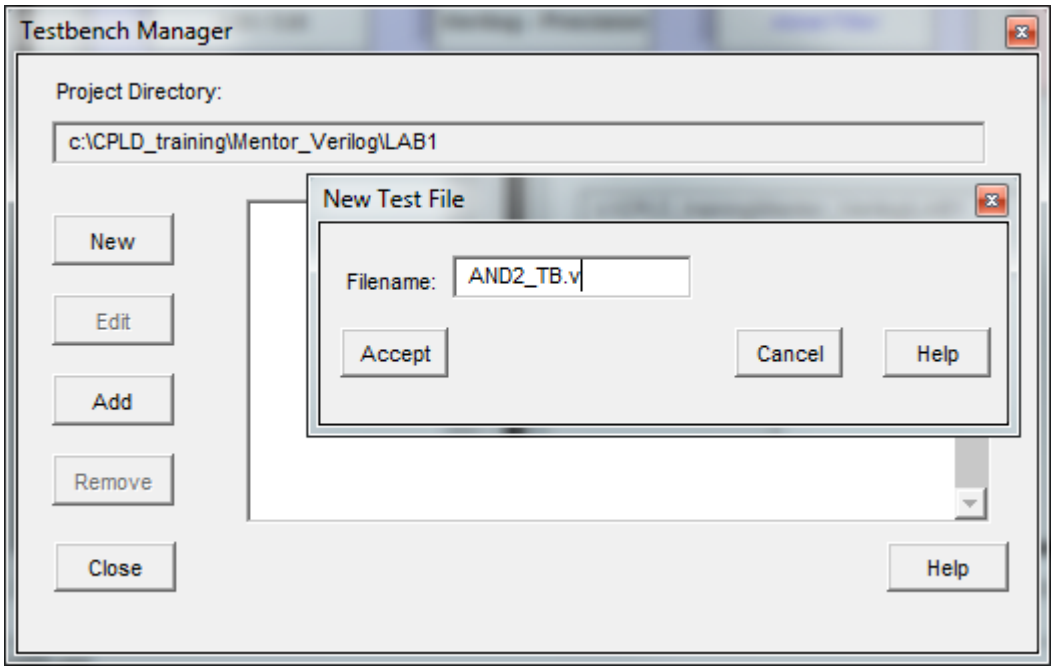
11. Go to the ***File*** menu in HDL Planner and then choose ***Exit*** to close HDL Planner.

Setup Testbench File for Simulation

Note:
To run VHDL/Verilog simulation, a separate license for ModelSim from Mentor Graphics is required. Please contact Mentor Graphics for details. If a ModelSim license is not available, please skip the simulation sections of the tutorial.

1. Press the ***Add / Edit*** button under **Testbench Manager** of the **Design Flow** tab window.

2. When the **Testbench Manager** window opens, click on the ***New*** button and then enter "**AND2_TB.v".** Then click on the ***Accept*** button and the **New module wizard** window will open**.**



3. In the **New module wizard** window, click ***Cancel*** to close this window. A blank file will open in HDL Planner.

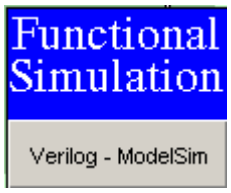
4. Add the following lines of code into the Verilog testbench template file.

```
`timescale 1ns/100ps
module AND2_TB;
// declare pin names
//begin
// add design description here

reg SIG_A, SIG_B;
wire   SIG_Q;
AND_2 U1(.A (SIG_A), .B(SIG_B), .Q(SIG_Q));
initial
begin
#0
SIG_A <= 1'b0;
SIG_B <= 1'b0;
#50
SIG_A <= 1'b0;
SIG_B <= 1'b1;
#50
SIG_A <= 1'b1;
SIG_B <= 1'b0;
#50
SIG_A <= 1'b1;
SIG_B <= 1'b1;
#50  ;
end

endmodule
```

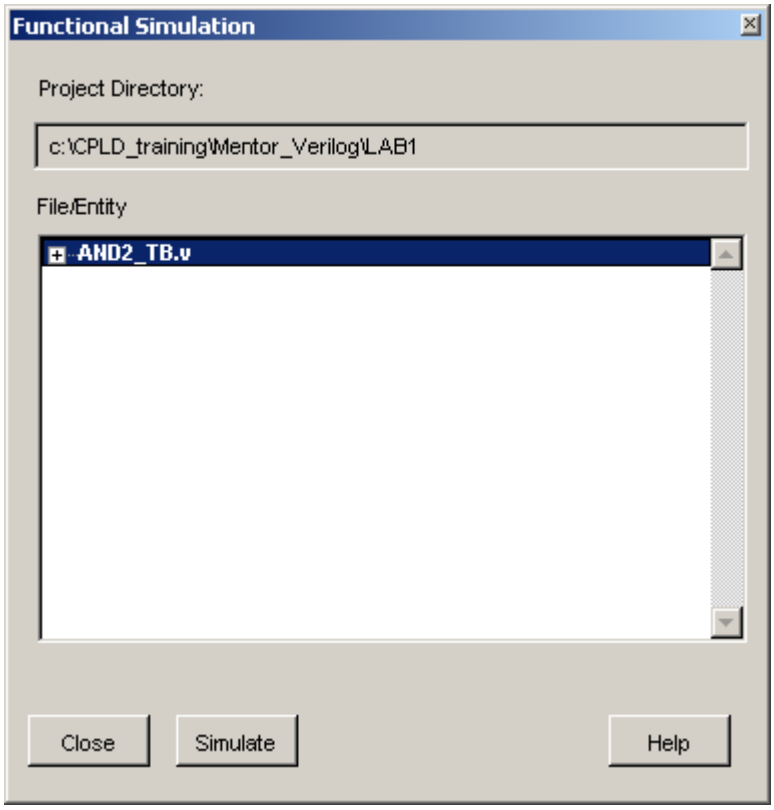
5. Go to the ***File*** menu of the HDL Planer and then select ***Save*** to save your Testbench file to the current project directory.
6. Go to the ***File*** menu of the HDL Planner and then choose ***Exit*** to close HDL Planner.

Functional Simulation using ModelSim

1. Click on the ***Verilog - ModelSim*** button under **Functional Simulation** of the **Design Flow** tab window.



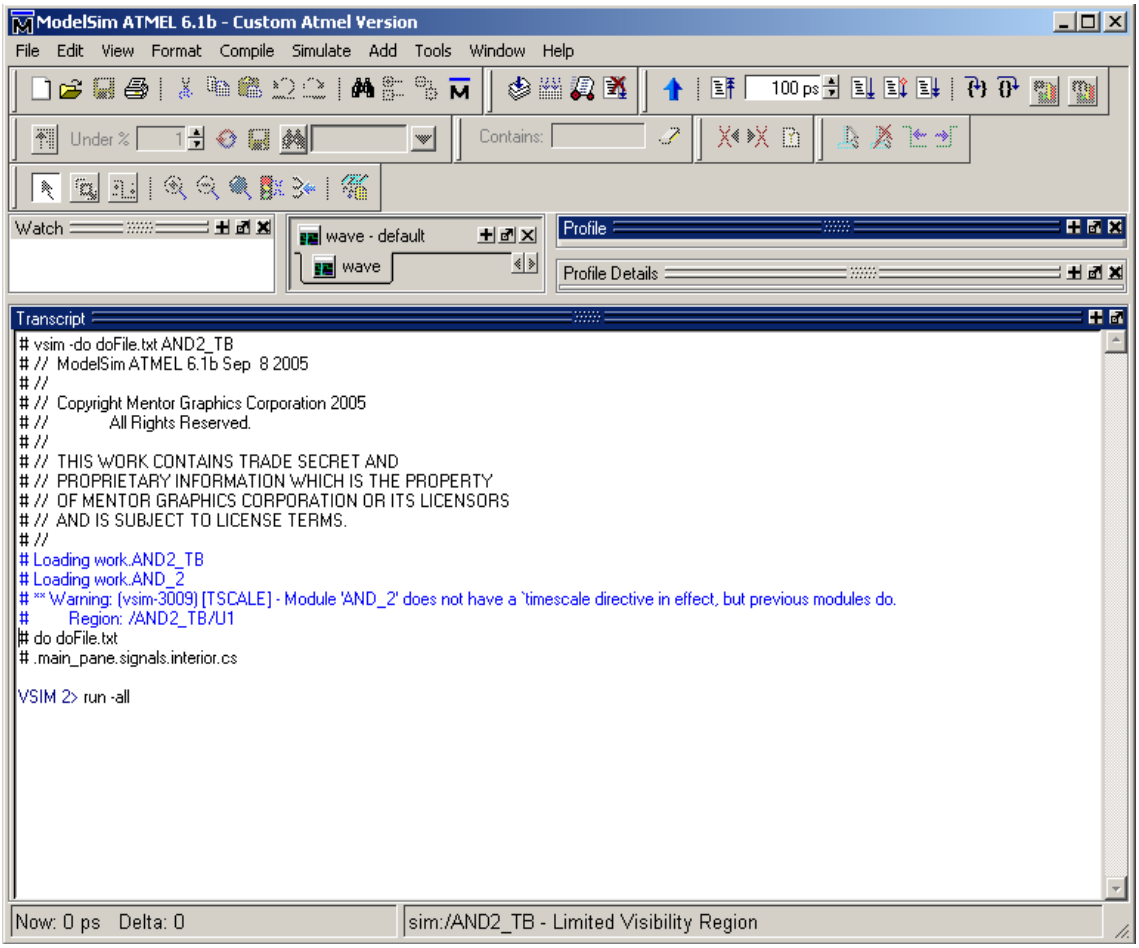
2. If there is no syntax error in the **AND2_TB.v** file, the following window will appear.



Note:
If the Log window at the bottom of the ProChip Designer window shows errors, please scroll up to see the actual error message in the Log window. Then, double-click on the Verilog file in the project tree to open the Verilog file in a text editor to correct the syntax error in the Verilog testbench file or top-level Verilog design file before performing functional simulation again.

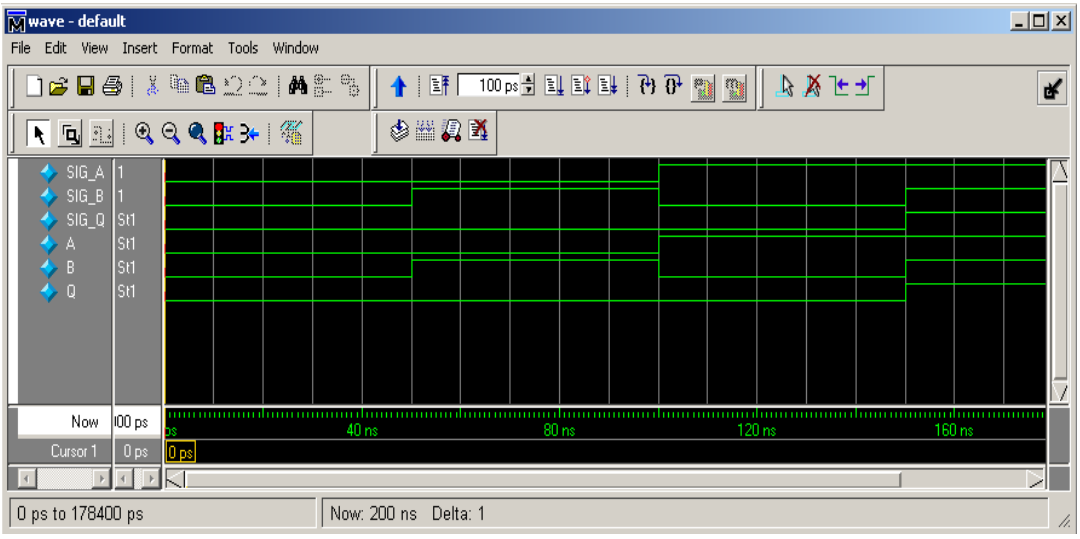
3. Highlight the testbench file ***AND2_TB.v*** in the **Functional Simulation** window and then press ***Simulate*** to open **ModelSim** to perform functional simulation.

4. Wait for ModelSim to launch. Then enter the ***run –all*** command in the **Transcript** window of **ModelSim** to run the entire simulation time specified in the testbench file.



5. In the main ModelSim window, go to ***Simulate*** ➔ ***Break*** to stop the simulation.

6. Use the ***Unlock*** button in the **Wave –default** window to maximize the **Wave** window.
7. Select ***View*** ➔ ***Zoom*** ➔ ***Zoom Full*** from the **Wave** window and then adjust the range for the displayed signals to see the complete waveform diagram as shown below.

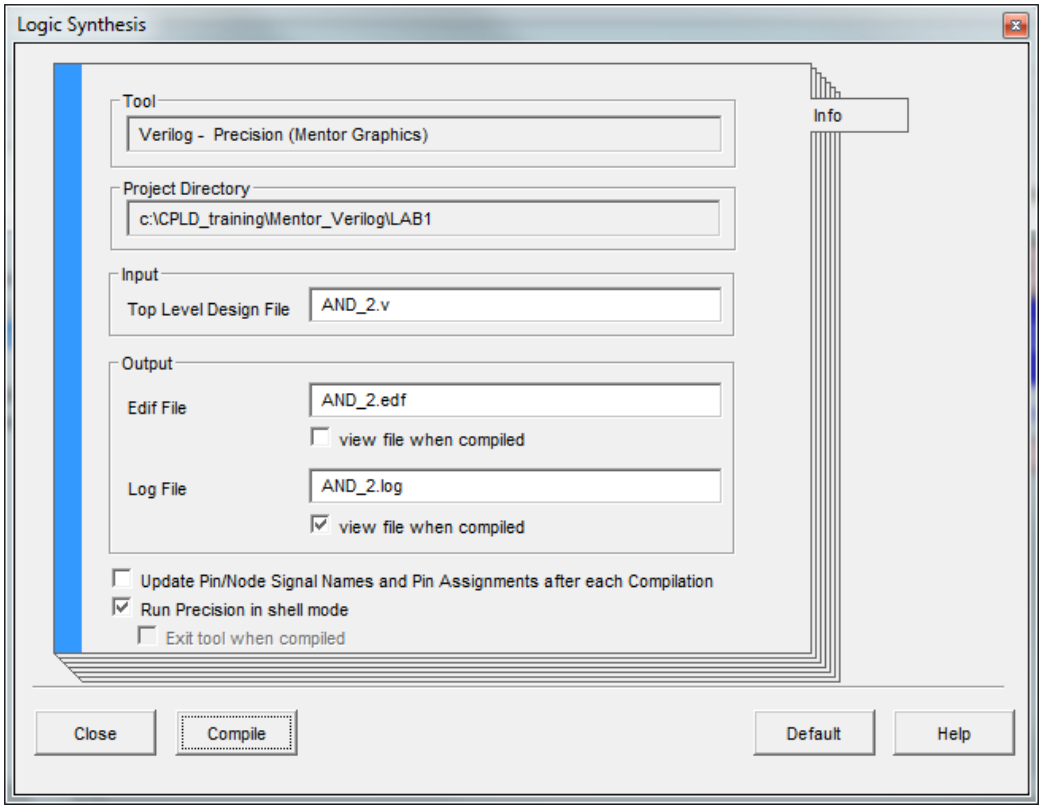


8. Select ***File*** ➔ ***Quit*** from **ModelSim** window and then select ***Yes*** to end simulation and close **ModelSim** window.

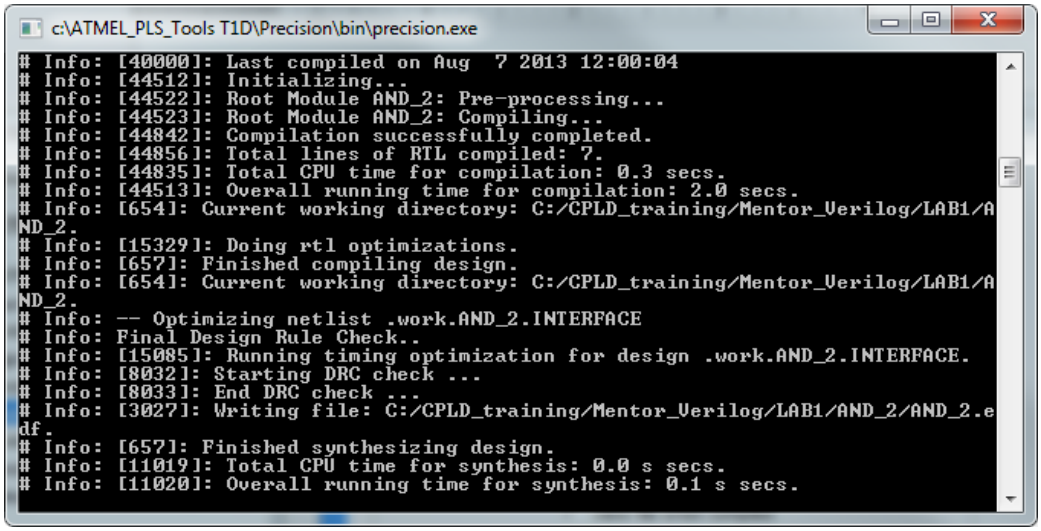
Logic Synthesis using Precision Synthesis

1. Press the ***Verilog - Precision*** button under **Logic Synthesis** of the Design Flow tab window.

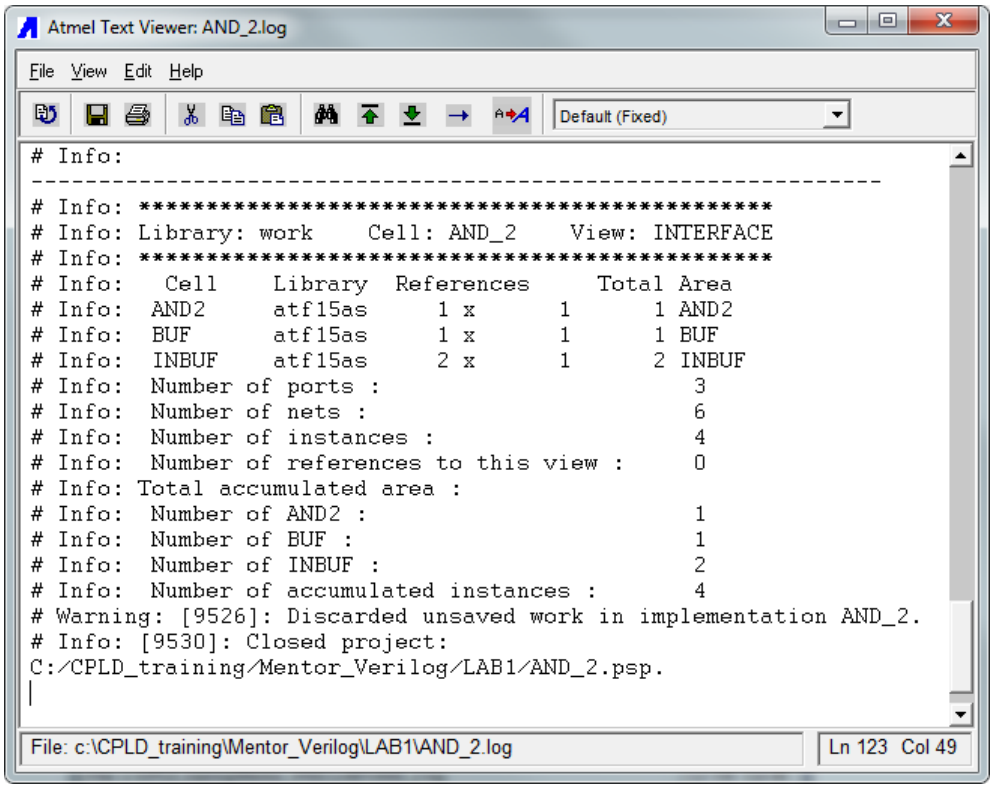
2. Click on the ***Compile*** button in the **Logic Synthesis** window to run Precision Synthesis.



Now, since the **Run Precision in Shell mode** option in the **Logic Synthesis** window is checked by default, ProChip Designer will display a DOS Prompt window when running the script file to compile and synthesize the design with Precision Synthesis.

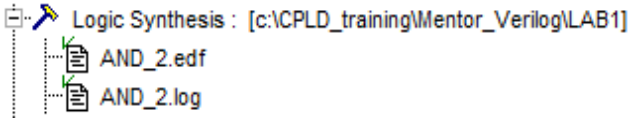


3. If there is no syntax error detected during compilation or synthesis of the design code, the **AND_2.log** file will appear. Review the **AND_2.log** file and then select ***File*** ➔ ***Exit*** to close it.



Note:
If syntax error is detected, you must go back to the text editor to correct the syntax error in the design code, and then start the synthesis process again.

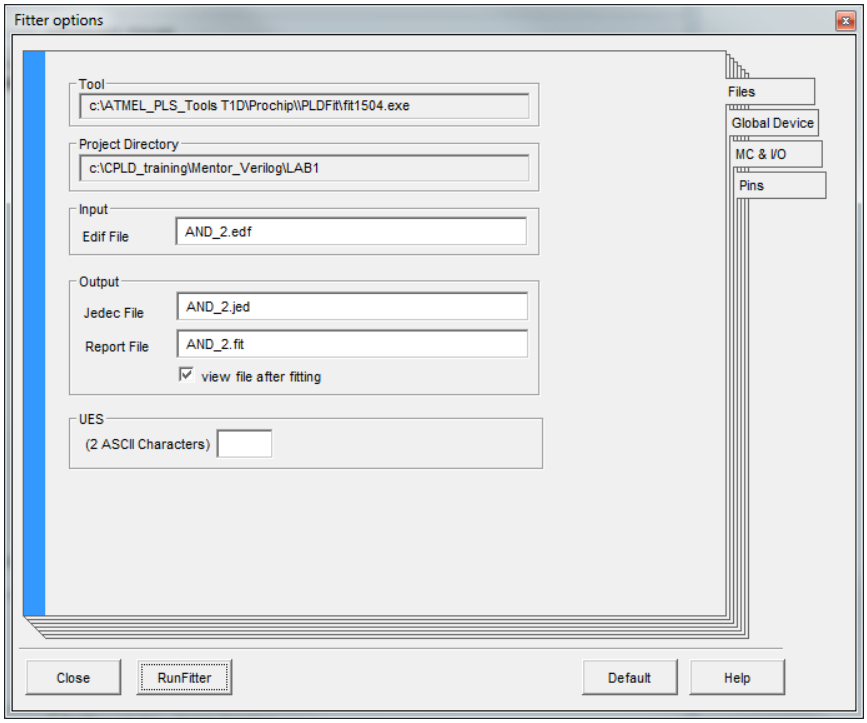
Now, **AND_2.edf** and **AND_2.log** are shown in the project tree as shown below.



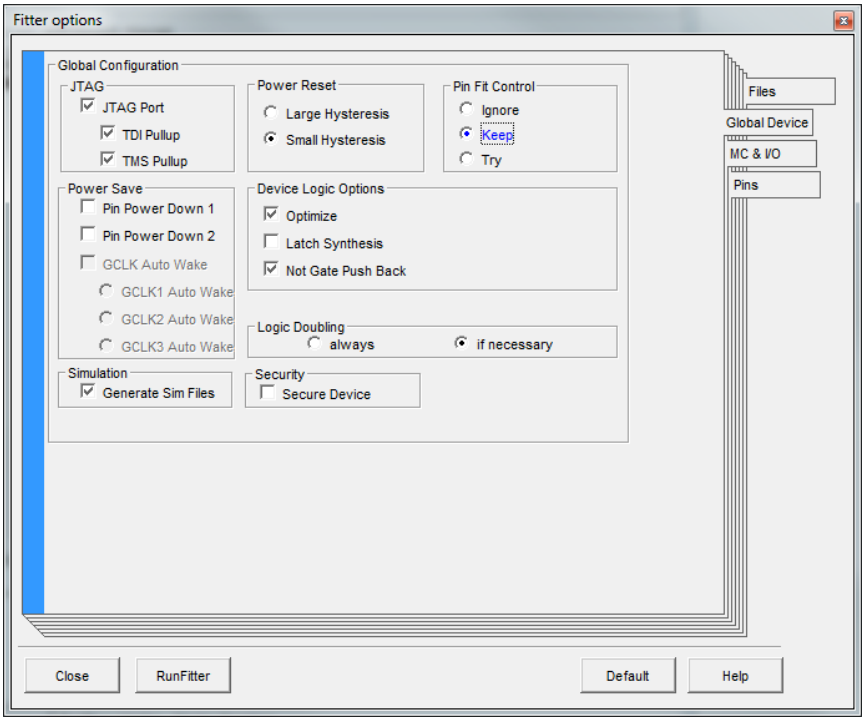
Device Fitting Using Atmel Fitter

1. Press the ***Atmel Fitter*** button under **Device Fitter** of the **Design Flow** tab window.

2. The **Fitter Options** window opens as shown below.



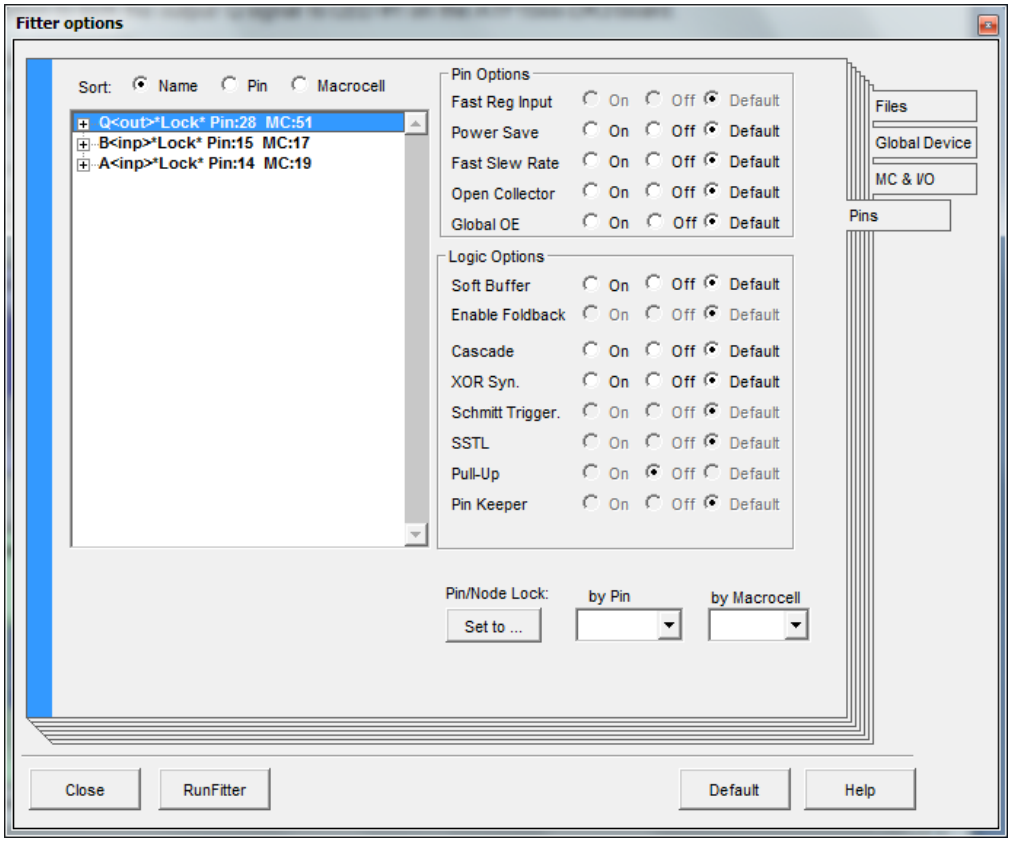
3. Select the ***Global Device*** tab.



4. Enable/check the ***Keep*** option for **Pin Fit Control** to keep the pre-assigned and locked input and I/O pin assignments. Pin assignments will be performed in the steps below.

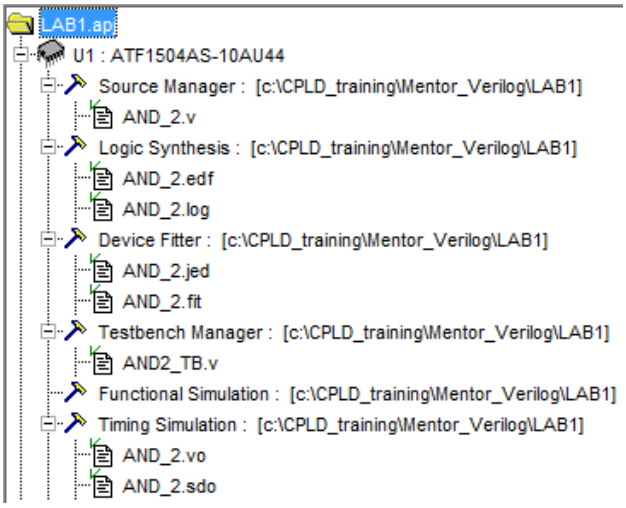
   Note:
   Please make sure that the **JTAG**, **TDI Pull-up**, and **TMS pull-up** options are enabled (checked), which are required to program the ATF15xx via JTAG ISP in later sections of this tutorial.

5. Select the ***Pins*** tab to see the input and output signals.
6. Highlight ***Q <out>*** in the window on the left and then select ***28*** under the **by Pin** drop-down menu to lock the output Q signal to LED1 on the ATF15xx-DK3 board.
7. Highlight ***B <inp>*** in the window on the left and then select **15** under the **by Pin** drop-down menu to lock the input B signal to SW1 push-button switch on the ATF15xx-DK3 board.
8. Highlight ***A <inp>*** in the window on the left and then select **14** under the **by Pin** drop-down menu to lock the input A signal to SW2 push-button switch on the ATF15xx-DK3 board.



9. Press the ***RunFitter*** button to fit the design and to generate a JEDEC programming file (.JED) for the selected device type. The Fitter Report file (.FIT) as well as the *.VO and *.SDO back-annotated simulation files will also be generated at the same time.
10. If there is no fitting issue, the following window will appear. Click ***OK*** to close it.

11. Click on the ***Close*** button to close the **Fitter Options** window. Now, you can review the **Fitter Report** file (**AND_2.fit**) for details of the fitted design.

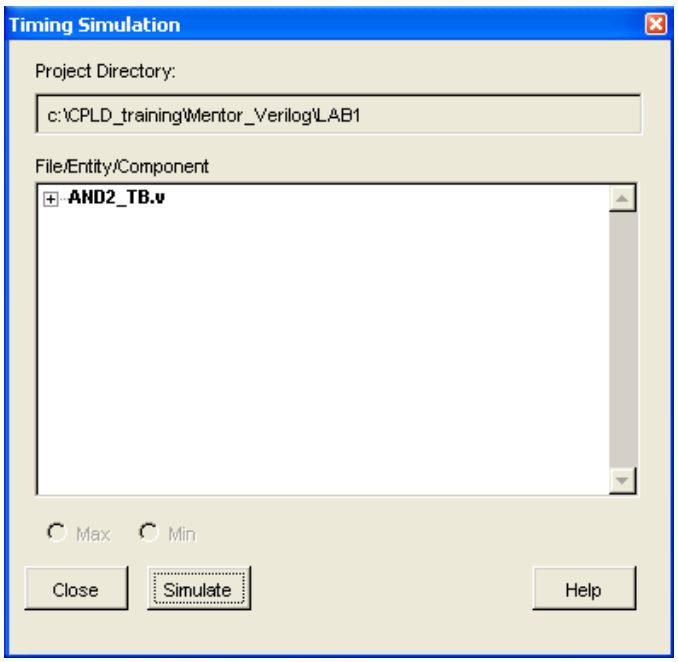


Note:
AND_2.vo and AND_2.sdo back-annotated simulation files must be present in order to perform timing simulation.

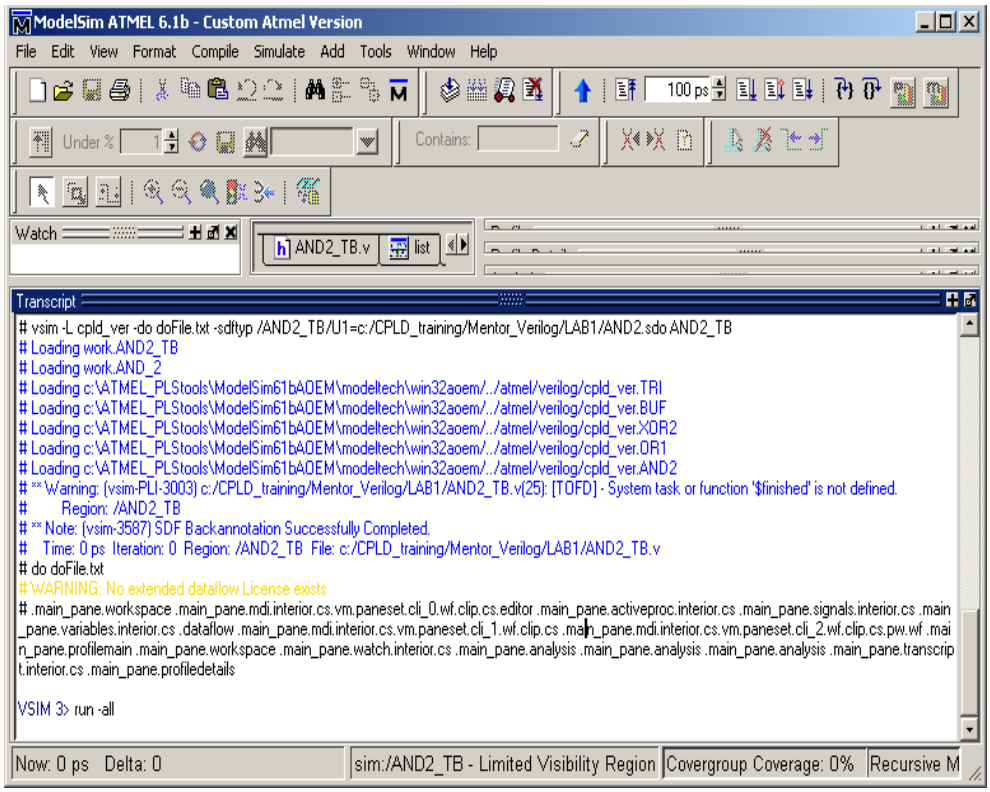
Timing Simulation using ModelSim

1. Since the Verilog testbench file was previously created for functional simulation and added to the Testbench Manager, you can now press the ***Verilog - ModelSim*** button under **Timing Simulation** of the **Design Flow** tab to run timing simulation on the AND_2 Verilog design.

2. Highlight ***AND2_TB.v*** in the **Timing Simulation** window and then click on the ***Simulate*** button to open ModelSim for timing simulation.



3. Wait for ModelSim to launch. Then enter the ***run –all*** command in the **Transcript** window of ModelSim to run the entire simulation time specified in the testbench file.

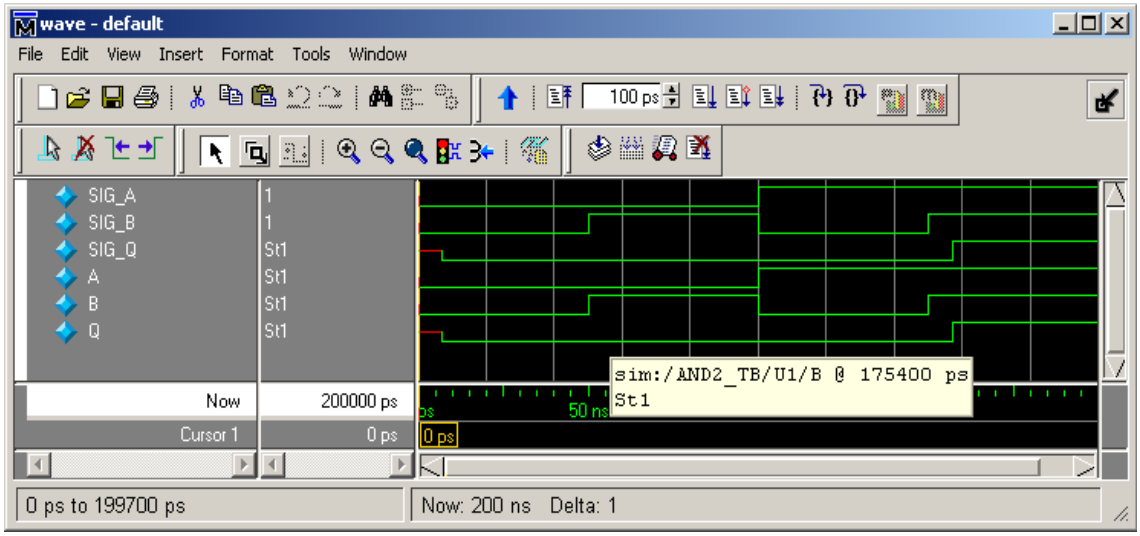


4. In the main ModelSim window, go to ***Simulate ➔ Break*** to stop the simulation.

5. Use the ***Unlock*** button in the **Wave –default** window to maximize the **Wave** window.

6. Select ***View ➔ Zoom ➔ Zoom Full*** from the **Wave** window and then use the mouse to adjust the range of the view area of the signal names to see them.

7. Select ***Edit ➔ Select All*** in the Wave window**.** Next, press and hold the ***Ctrl*** key and then click on the ***SIG_A, SIG_B, SIG_Q, A, B,*** **and** ***Q*** signals one at a time to de-select these signals. Select ***Edit ➔ Delete*** to remove all other signals except ***SIG_A, SIG_B, SIG_Q, A, B,*** **and** ***Q***. The final waveforms will be displayed as shown below.



8. Select ***File*** ➔ ***Quit*** in the **Wave** window and then select ***Yes*** to end simulation and close ModelSim.

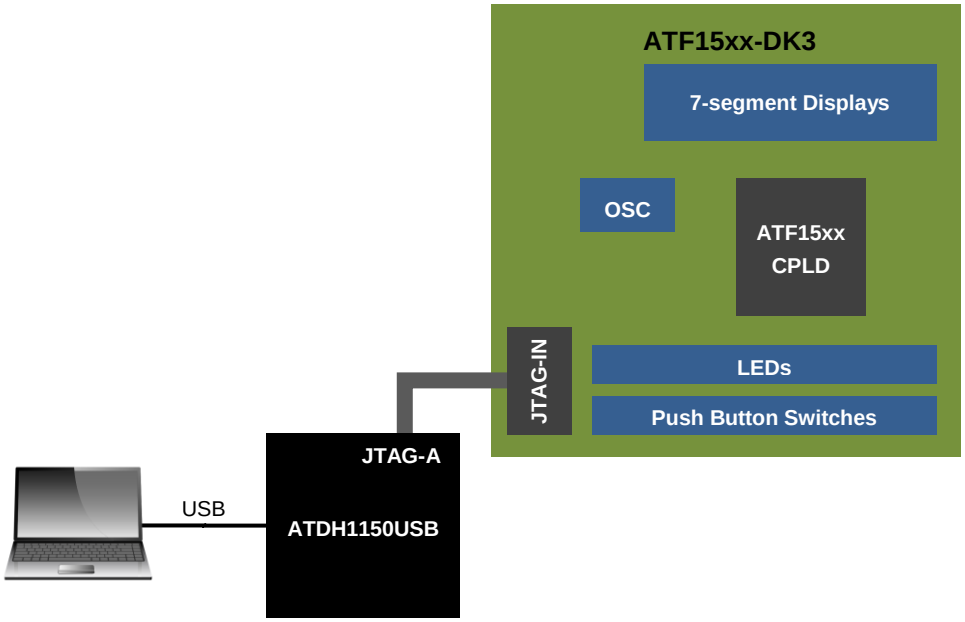
In-System Programming using ATF15xx-DK3-U and ATMISP

Hardware Setup:

1. Connect the **USB cable** contained in the **ATDH1150USB** kit to the **USB port** of the computer and the **USB connector** of the **ATDH1150USB** cable.
2. Connect the **10-wire ribbon cable** contained in the kit to the **JTAG-A** port of the **ATDH1150USB** and to the JTAG header (**JTAG-IN**) of the **ATF15xx-DK3** board.

   Note:
   The selection jumper at **JP-TDO** on the **ATF15xx-DK3** board should be plugged into the "**TO ISP CABLE**" side.
3. Insert the 44-TQFP socket adapter board (**ATF15xxDK3-SAA44**) onto main **ATF15xx-DK3** board if it is not already inserted.
4. Set both the **VccIO** and **VccINT** selection jumpers on the ATF15xx-DK3 board to ***5V*** if an **ATF15xxAS/ASL** device is being used. If an **ATF15xxASV/ASVL** device is being used, set the **VccIO** and **VccINT** selection jumpers on the ATF15xx-DK3 board to ***3.3V***.

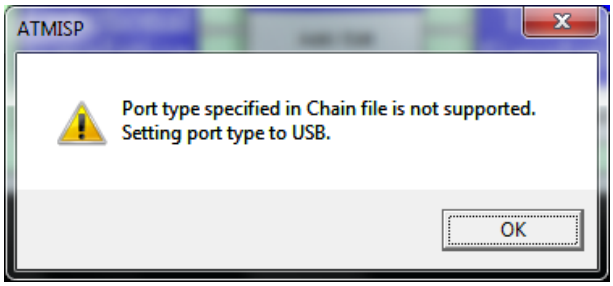
   Note:
   The **Power Switch** for the **ATF15xx-DK3** board must be turned ***OFF*** before changing the positions of the **VccIO** and **VccINT** selection jumpers.
5. Insert a blank **ATF1504AS-10AU44** (or the device type selected in the ProChip Designer project) into the 44-pin TQFP socket. Please note that pin 1 of the device should be at the upper left hand corner of the socket facing the U1 label.
6. Set ***JPL1*** jumper to use **LED1**, **JPS1** jumper to use the first Push-button switch (**SW1**), and JPS2 to use the second Push-button switch (**SW2**).
7. Connect the **9V DC** power source to the power connector at **JPower** of the **ATF15xx-DK3** board.
8. Turn the power on by bringing the **Power Switch** of the **ATF15xx-DK3** board to the ***ON*** position.

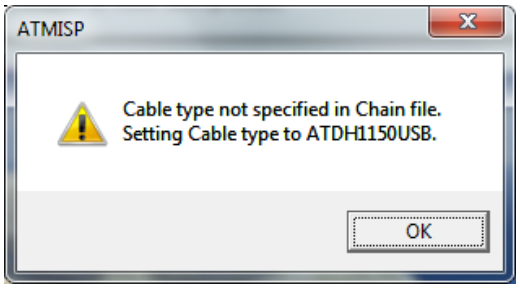


Software Setup:

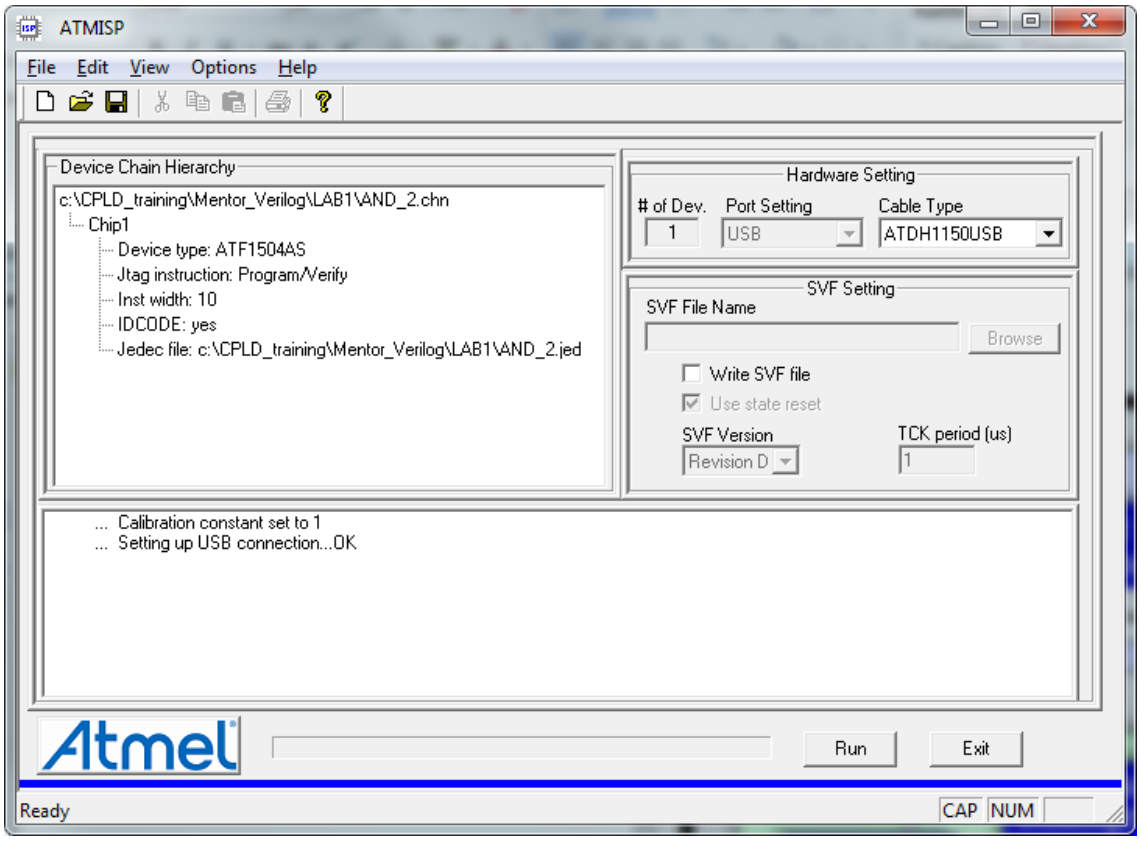
1. In ProChip Designer, press the ***Program Chip*** button under **Atmel-ISP** of the **Design Flow** tab window.

2. Click ***OK*** on the two **ATMISP** warning message prompts about the port and cable types that pop up when ATMISP is being launched. ATMISP v7.x only supports the USB port type and ATDH1150USB cable type.





3. ATMISP opens and automatically loads the Chain file (.CHN) created by ProChip Designer for the current design.



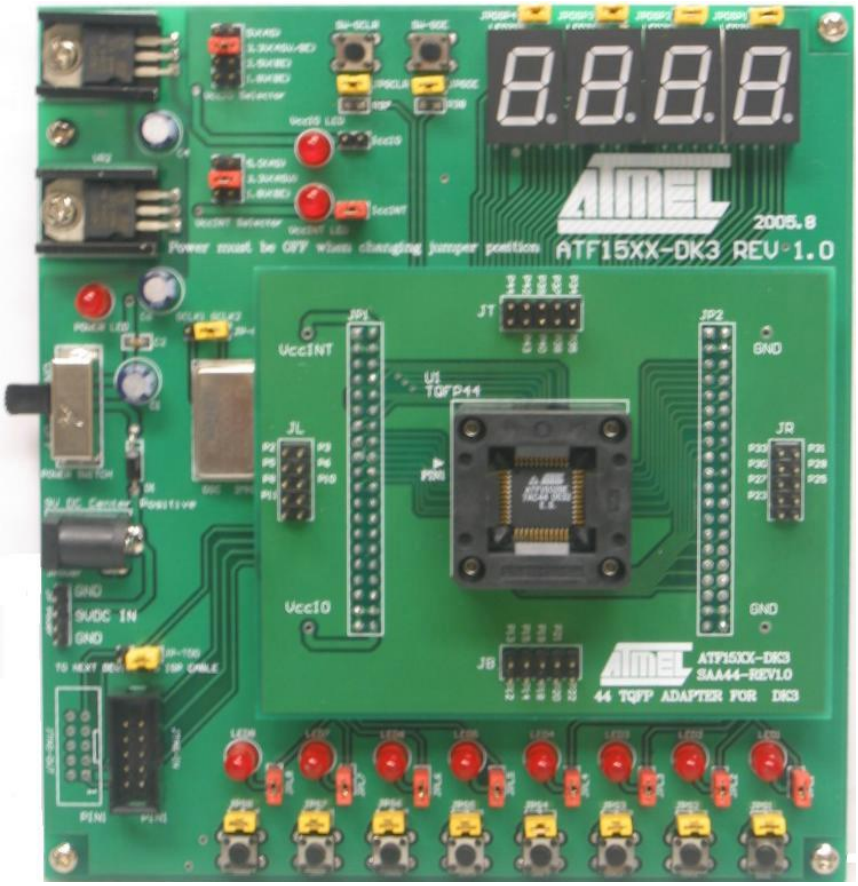
4. Press the ***Run*** button in the **ATMISP** window to start the JTAG In-system programming.

5. When programming is successfully done, press ***OK*** and then select **File ➔ Exit** to close ATMISP software.

6. Select ***Project ➔ Save***, and then ***Project ➔ Exit*** in Prochip Designer to save the design and then exit the software.

Testing Design on ATF15xx-DK3 Board

1. Press and then hold the ***SW1*** push-button switch (**B**) and ***SW1*** push-button switch (**A**) at the same time to see the **LED1** result (`assign Q = A & B;`) on the ATF15xx-DK3 board.



END OF TUTORIAL 1

Tutorial 2: 2-bit Full Adder Design

Create a New Project in ProChip Designer

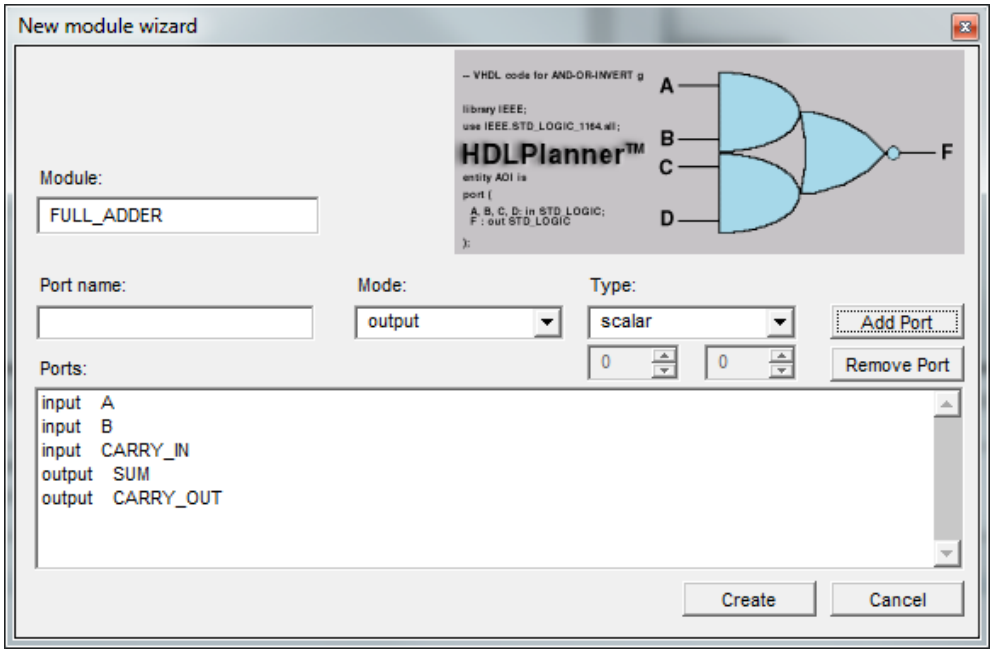
1. Double click on the ***ProChip Designer 5.0*** icon to launch ProChip Designer.
2. Under the ***Project*** menu of ProChip Designer, select ***New Project Wizard…***
3. In the **New Project Wizard – Step 1 of 6** window, click on the ***Next >*** button.
4. In **New Project Wizard – Step 2 of 6** window, click on the ***Browse*** button and create a new design directory "**C:\CPLD_training\Mentor_Verilog\LAB2"** and specify "**LAB2.apj**" as the project name. Then click on the ***Next >*** button.
5. In the **Part Number** dialog box of the **New Project Wizard – Step 3 of 6** window, select ***ATF1504AS-10AU44*** (or any ATF15xx 44-TQFP that is available to you) as the target device type. Then click on the ***Next >*** button.
6. In the **Tool flow** dialog box of the **New Project Wizard – Step 4 of 6** window, select ***Verilog - Mentor Graphics***. Then click on the ***Next >*** button.
7. In the **New Project Wizard – Step 5 of 6** window, click on the ***Next >*** button.
8. In the **New Project Wizard – Step 6 of 6** window, click on the ***Finish*** button to close the **New Project Wizard**.

Create a New Verilog Design Using HDL Planner

1. Click on the ***Add / Edit*** button under **Source Manager** of the **Design Flow** tab window.

2. Click on the ***New*** button in the **Source Manager** window to open the **New Design File** dialog window.
3. Enter the Verilog design filename "**FULL_ADDER.v**" into the **New Design File** dialog window and then click on the ***Accept*** button. The **New module wizard** window will open.
4. In the **New module wizard** window, enter "**FULL_ADDER**" into the **Entity name** box.
5. Enter "**A**" into the **Port name** box, select ***Input*** under **Mode** and ***scalar*** under **Type**, and then click on the ***Add Port*** button.
6. Enter "**B**" into the **Port name** box, select ***Input*** under **Mode** and ***scalar*** under **Type**, and then click on the ***Add Port*** button.
7. Enter "**CARRY_IN**" into the **Port name** box, select ***Input*** under **Mode** and ***scalar*** under **Type**, and then click on the ***Add Port*** button.
7. Enter "**SUM**" into the **Port name** box, select ***Output*** under **Mode** and ***scalar*** under **Type,** and then click on the ***Add Port*** button.
8. Enter "**CARRY_OUT**" into the **Port name** box, select ***Output*** under **Mode** and ***scalar*** under **Type,** and then click on the ***Add Port*** button.

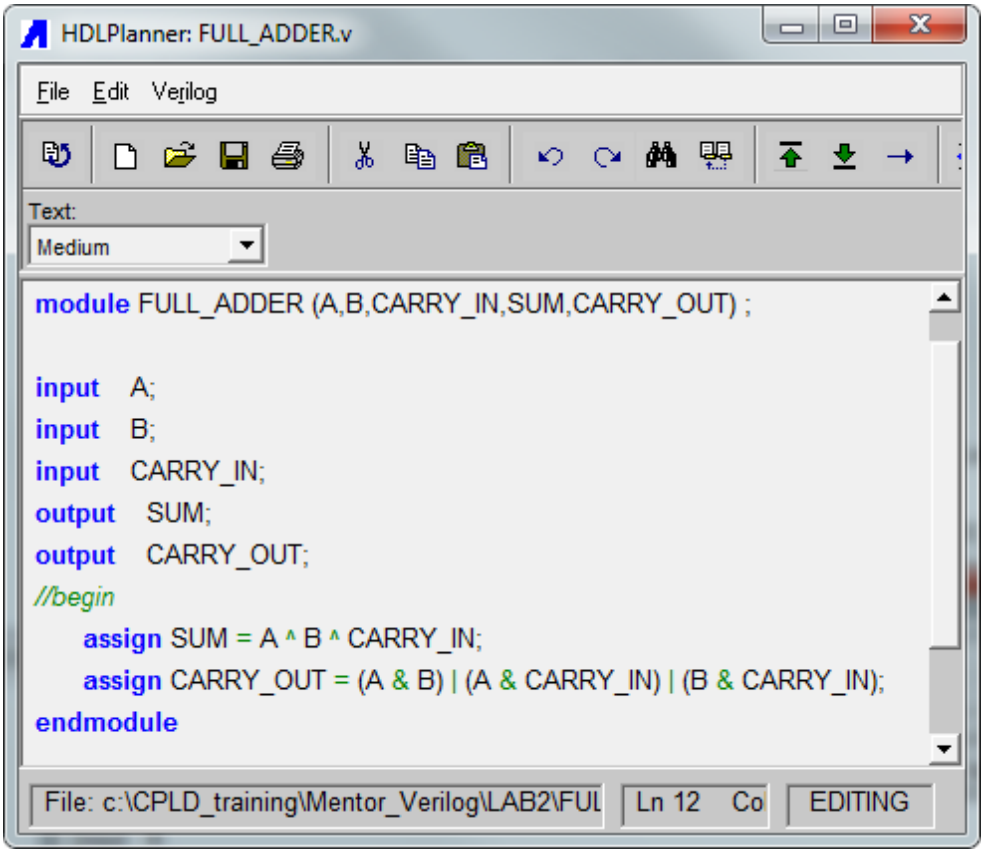
Now, the **Ports** dialog box will show the input and output ports as shown below.



9. Click on the ***Create*** button to create the Verilog design file with the Entity, Input Ports and Output Ports specified in the **New module wizard** and the Verilog design file will open in HDL Planner.

10. Add the following logic equation to the Verilog design file before `endmodule`:

```
assign SUM = A ^ B ^ CARRY_IN;
assign CARRY_OUT = (A & B) | (A & CARRY_IN) | (B & CARRY_IN);
```



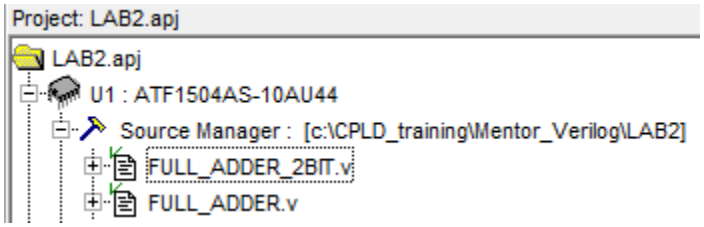
11. Go to the ***File*** menu in HDL Planer and then select ***Save*** to save your design to the current project directory.
12. Go to the ***File*** menu in HDL Planner and then choose ***Exit*** to close HDL Planner.
13. Click on the ***Add / Edit*** button under **Source Manager** of the **Design Flow** tab window again to start creating the top-level design file.
14. Click on the ***New*** button in the **Source Manager** window to open the **New Design File** dialog window to create another new design file.
15. Enter the Verilog design filename "**FULL_ADDER_2BIT.v**" into the **New Design File** dialog window and then click on the ***Accept*** button. The **New module wizard** window will open.
16. Click on the ***Cancel*** button in the **New module wizard** window and then create the following top-level Verilog design code in HDL Planner directly.

    You can use the **Select Tool** feature in the PDF reader to select the following code in this tutorial file and then copy-and-paste it to HDL Planner.

```
module FULL_ADDER_2BIT (X, Y, CIN, COUT, SUM);
input [1:0] X;
input [1:0] Y;
input CIN;
output COUT;
output [1:0] SUM;
wire C1;

FULL_ADDER U0( .A(X[0]), .B (Y[0]),  .CARRY_IN(CIN),
               .CARRY_OUT (C1), .SUM(SUM[0]));

FULL_ADDER U1( .A(X[1]),   .B (Y[1]),  .CARRY_IN(C1),
               .CARRY_OUT (COUT), .SUM (SUM[1]));
endmodule
```

Note:
Verilog is case-sensitive language. So please make sure to use upper or low case characters accordingly.

17. Go to the ***File*** menu of the HDL Planner and then select ***Save*** to save the top-level design file to the current project directory.
18. Go to the ***File*** menu of the HDL Planner and then choose ***Exit*** to close HDL Planner.
19. If FULL_ADDER_2BIT.v is not already on top of FULL_ADDER.v in the ProChip Designer **Project** window, select the **FULL_ADDER_2BIT.v** file in the **Project** window and click & hold the mouse to move it above **FULL_ADDER.v** file since the top-level design file must be placed on the top of other design files.



Setup Testbench File for Simulation

Note:
To run VHDL/Verilog simulation, a separate license for ModelSim from Mentor Graphics is required. Please contact Mentor Graphics for details. If a ModelSim license is not available, please skip the simulation sections of the tutorial.

1. Press the ***Add / Edit*** button under **Testbench Manager** of the **Design Flow** tab window.

2. When the **Testbench Manager** window opens, click on the ***New*** button and then enter "**FULL_ADDER_2BIT_TB.v".** Then click on the ***Accept*** button and the **New module wizard** window will open**.**
3. In the **New module wizard** window, click ***Cancel*** to close this window. A blank file will open in HDL Planner.
4. Add the following lines of code into the Verilog testbench template file.

   You can use the **Select Tool** feature in the PDF reader to select the following code in this tutorial file and then copy-and-paste it to HDL Planner.

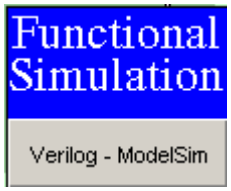
```
`timescale 1ns/100ps

 module FULL_ADDER_2BIT_TB;

 reg [1:0] SIG_X;
 reg [1:0] SIG_Y;
 reg SIG_CIN;
 wire SIG_COUT;
 wire [1:0] SIG_SUM;

// declare pin names
// begin
// add design description here

FULL_ADDER_2BIT U11( .X (SIG_X), .Y (SIG_Y), .CIN(SIG_CIN),
                     .COUT(SIG_COUT), .SUM(SIG_SUM) );

initial
begin

#0
SIG_X  <= XY /*Syntax error here, intentionally placed here. It should be
"2'b00;"*/
 SIG_Y  <= 2'b00;
 SIG_CIN <= 1'b0;

 #10
 SIG_X  <= 2'b00;
 SIG_Y <=  2'b00;
 SIG_CIN <= 1'b1;

 #10
```

```
SIG_X  <= 2'b00;
SIG_Y <=  2'b01;
SIG_CIN <= 1'b0;

#10
SIG_X  <= 2'b00;
SIG_Y <= 2'b01;
SIG_CIN <= 1'b1;

#10
SIG_X  <= 2'b01;
SIG_Y <=  2'b01;
SIG_CIN <= 1'b0;

#10
SIG_X  <= 2'b11;
SIG_Y <= 2'b11;
SIG_CIN <= 1'b1;

#10
SIG_X  <= 2'b10;
SIG_Y <= 2'b10;
SIG_CIN <= 1'b1;

#10
SIG_X  <= 2'b11;
SIG_Y <= 2'b10;
SIG_CIN <= 1'b1;

#10 ;
end
endmodule
```

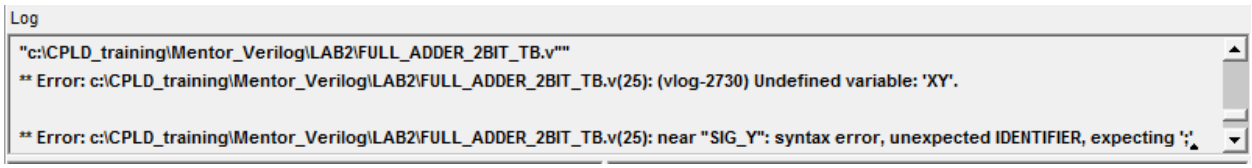
5. Go to the ***File*** menu of the HDL Planer and then select ***Save*** to save your Testbench file to the current project directory.
6. Go to the ***File*** menu of the HDL Planner and then choose ***Exit*** to close HDL Planner.

Functional Simulation using ModelSim

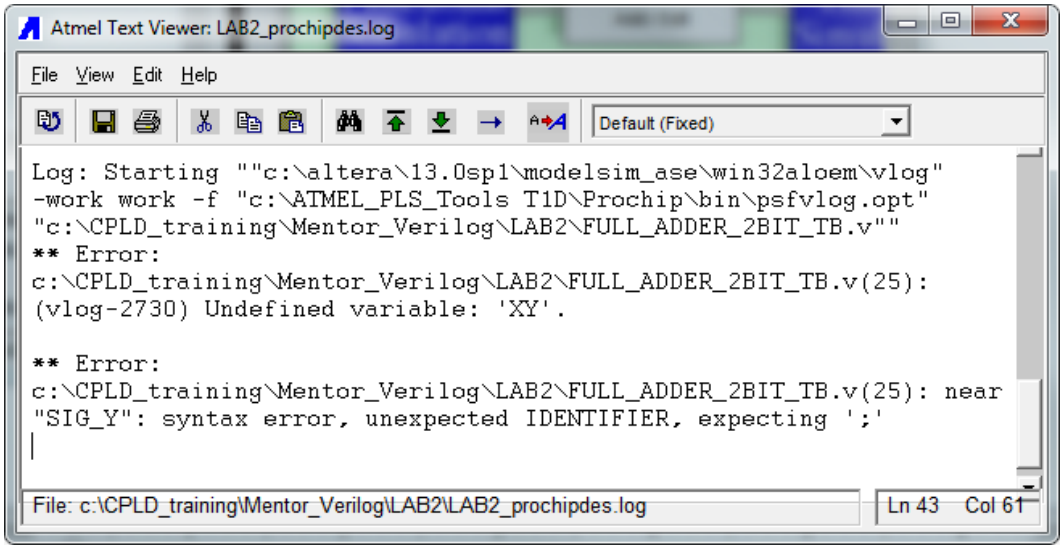
1. Click on the ***Verilog - ModelSim*** button under **Functional Simulation** of the **Design Flow** tab window.



2. Use the mouse to expand the **Log** window and then scroll up to see the syntax error message. Please note that the syntax error for the signal "**XY**" was intentionally entered to the testbench code.

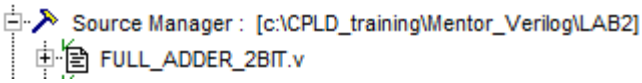


3. Click on the ***Open Log Viewer*** button in the vertical toolbar to open the log file and review the error messages in the log file again.

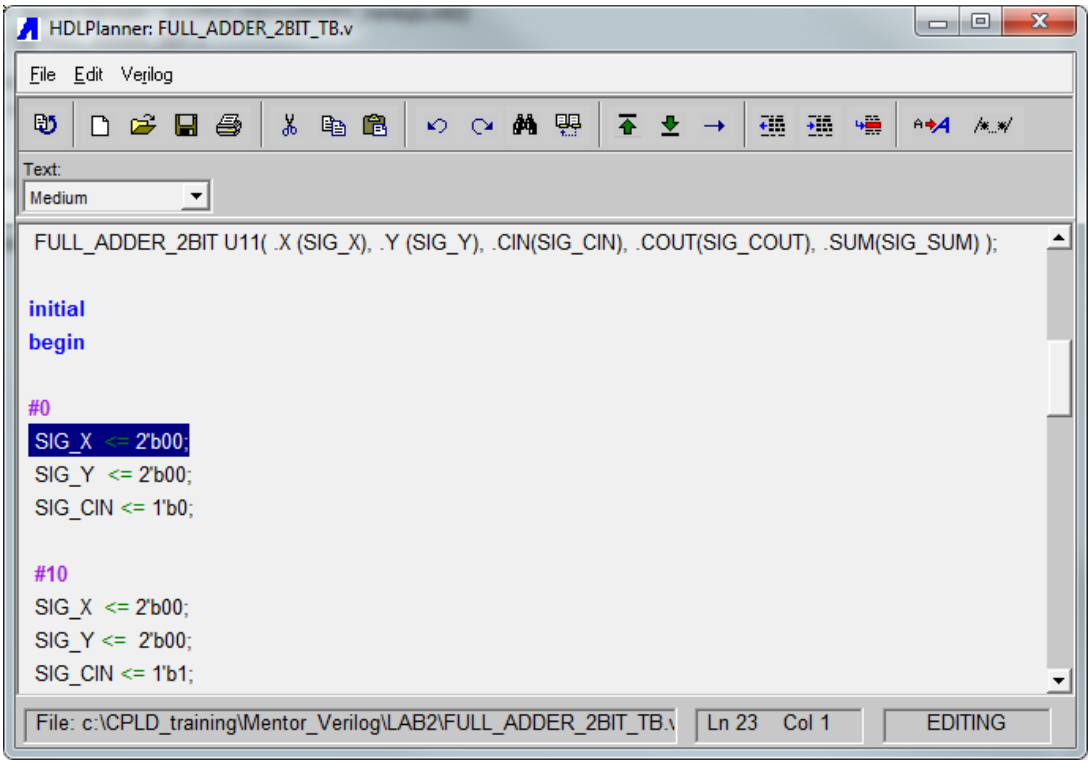


4. In the Log file window, select ***File*** ➔ ***Exit*** to close the log file.

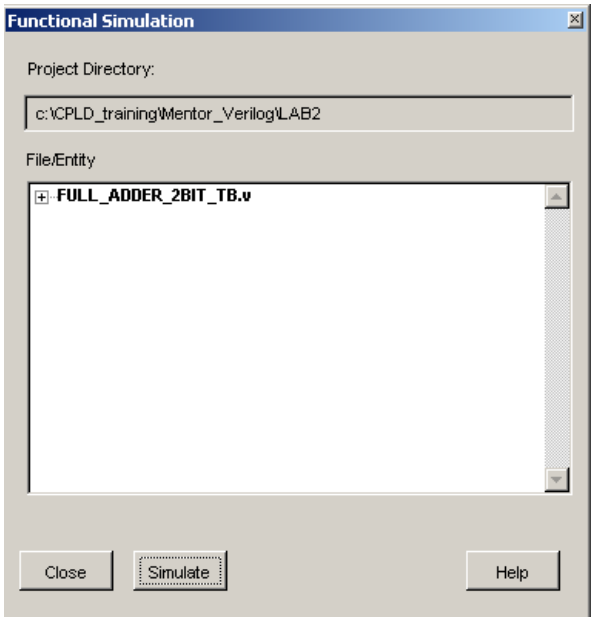
5. In the **Project** window, double click on the ***FULL_ADDER_2BIT_TB.v*** file located under **Testbench Manager** to edit the file using HDL Planner.



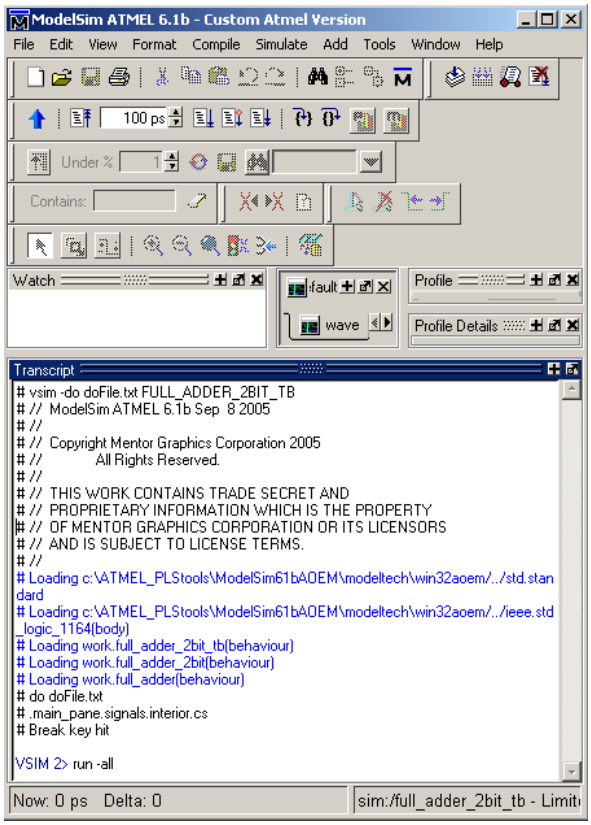
6. Change "**`XY`**" in the Verilog code to "**`2'b00;`**" as shown below to correct the syntax error in the specific line of the **FULL_ADDER_2BIT_TB.v** file in HDL Planner.



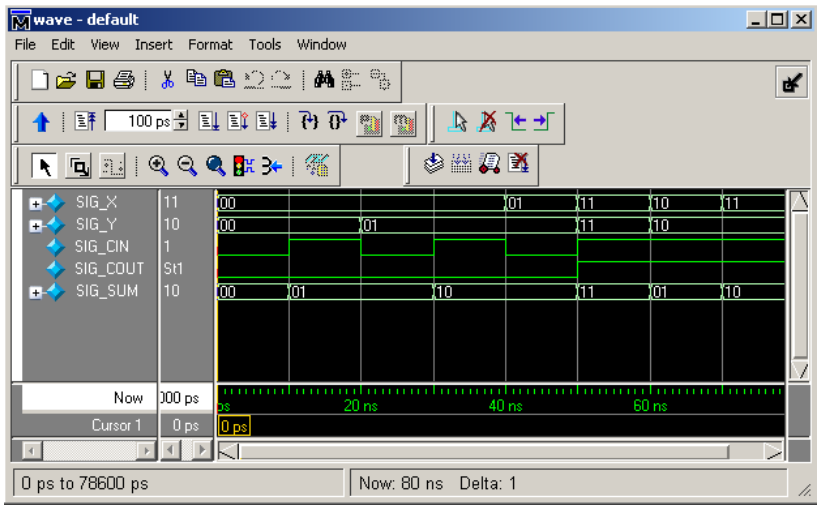
7. Select ***File*** ➔ ***Save*** and then ***File*** ➔ ***Exit*** to close HDL Planner.
8. Click on the ***Verilog - ModelSim*** button under **Functional Simulation** of the **Design Flow** tab window to perform functional simulation again.
9. Highlight the testbench file ***FULL_ADDER_2BIT_TB.v*** in the **Functional Simulation** window and then press ***Simulate*** to open ModelSim to perform functional simulation.



10. Wait for ModelSim to launch. Then enter the ***run –all*** command in the **Transcript** window of ModelSim to run the entire simulation time specified in the testbench file.



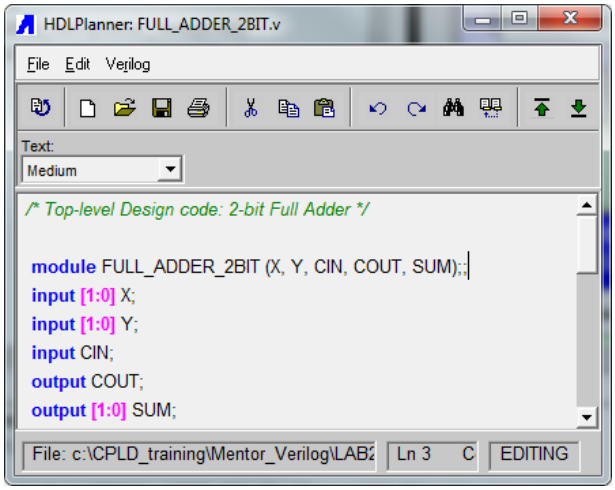
11. In the main ModelSim window, go to ***Simulate*** ➔ ***Break*** to stop the simulation.
12. Use the ***Unlock*** button in the **Wave –default** window to maximize the **Wave** window.
13. Select ***View*** ➔ ***Zoom*** ➔ ***Zoom Full*** from the **Wave** window and then adjust the range for the displayed signals to see the complete waveform diagram as shown below.
14. Select ***Edit*** ➔ ***Select All*** in the **Wave** window.
15. Press and hold the ***Ctrl*** key and then click on the ***SIG_X, SIG_Y, SIG_CIN, SIG_COUT,*** and ***SIG_SUM*** signals one at a time to de-select these signals. Select ***Edit*** ➔ ***Delete*** to remove all other signals. The final waveforms will be displayed as shown below.
16. Click in the ***middle of the wave window*** and then use the ***zoom out*** tool to obtain a better view of the waveform diagram as shown below.



17. Select ***File*** ➔ ***Save*** in the **Wave** window to save the waveform into **"C:/CPLD_training/Mentor_Verilog/LAB2/wave.do".**
18. Select ***File*** ➔ ***Quit*** in the **Wave** window and then select ***Yes*** to end simulation and close ModelSim.

Intentionally Create Syntax Error in Verilog Design File

1. Double click on the ***FULL_ADDER_2BIT.v*** file in the **Project** window to open it in HDL Planner.
2. Add "`;`" after the "`module FULL_ADDER_2BIT (X, Y, CIN, COUT, SUM);`" statement in the top-level design file to intentionally create a syntax error.

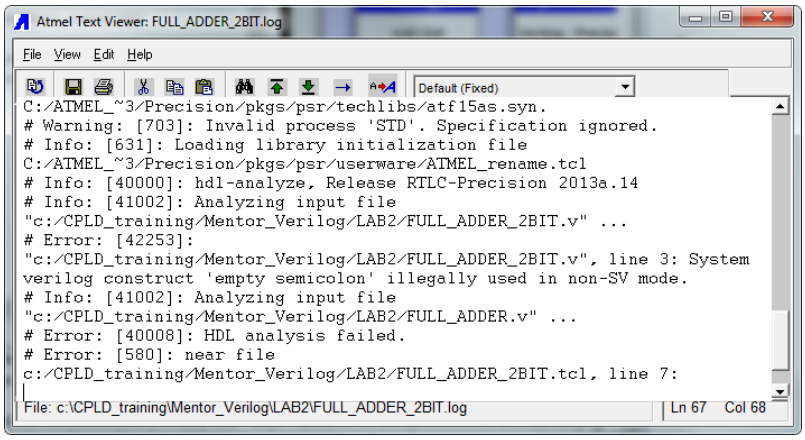


3. Go to ***File*** ➔ ***Save*** in HDL Planner to save the modified top-level design file.
4. Go to ***File*** ➔ ***Exit*** in HDL Planner to to close HDL Planner.

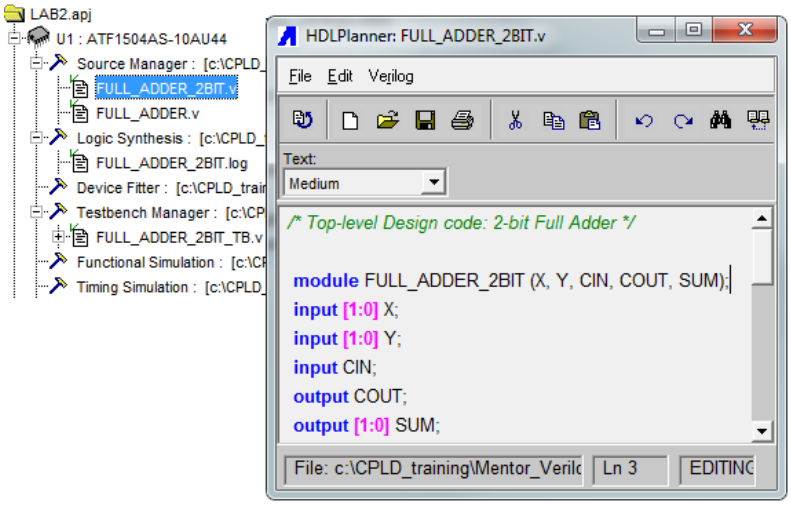
Logic Synthesis using Precision Synthesis

1. Press the ***Verilog - Precision*** button under **Logic Synthesis** of the Design Flow tab window.

2. Click on the ***Compile*** button in the **Logic Synthesis** window to run Precision Synthesis.
3. At the end of the synthesis process, the Log file will be displayed showing a syntax error in line 3 of FULL_ADDER_2BIT.v which is caused by the extra "**;**" added in the previous section. Select ***File*** ➔ ***Exit*** to close the log file.



4. Double click on the ***FULL_ADDER_2BIT.v*** file in the **Project** window to open it in HDL Planner.
5. Remove the extra ";" added into **FULL_ADDER_2BIT.v** in the previous section.



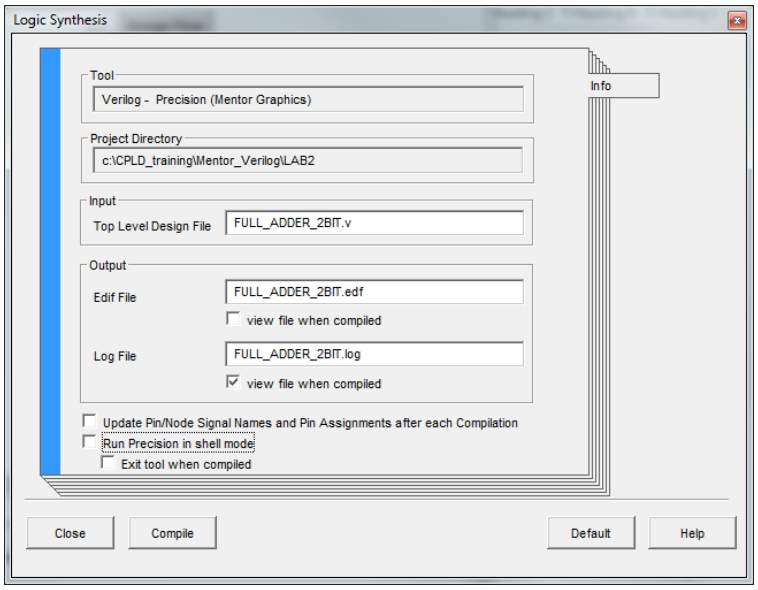
5. In HDL Planner, go to ***File*** ➔ ***Save*** to save the file and then ***File*** ➔ ***Exit*** to close the HDL planner.
6. Press the ***Verilog - Precision*** button under **Logic Synthesis** of the Design Flow tab window.

7. Click on the ***Compile*** button in the **Logic Synthesis** window to compile and synthesis the design again.

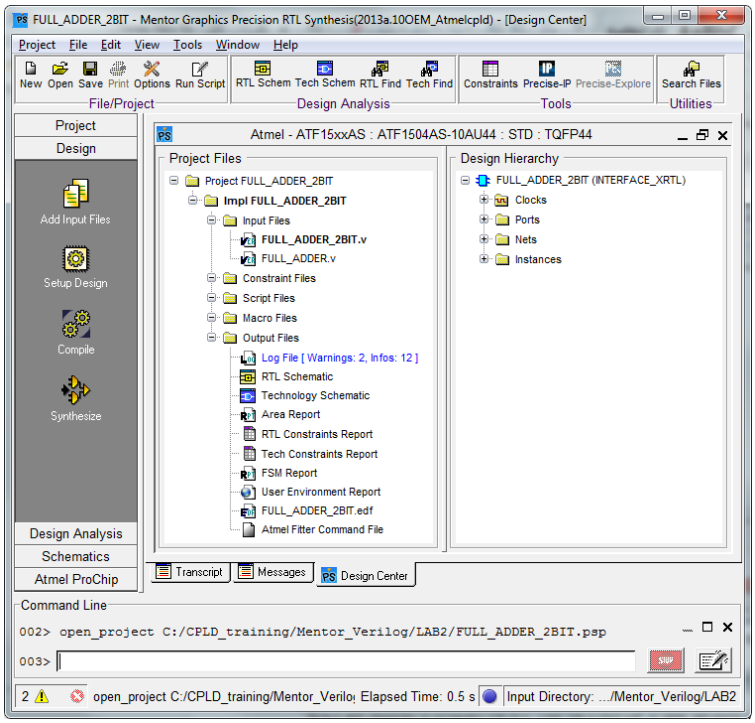
8. If there is no syntax error detected during compilation or synthesis of the design code, the **FULL_ADDER_2BIT.log** file will appear. Review the **FULL_ADDER_2BIT.log** file and then select ***File ➔ Exit*** to close the Log file.

9. Press the ***Verilog - Precision*** button under **Logic Synthesis** of the Design Flow tab window.

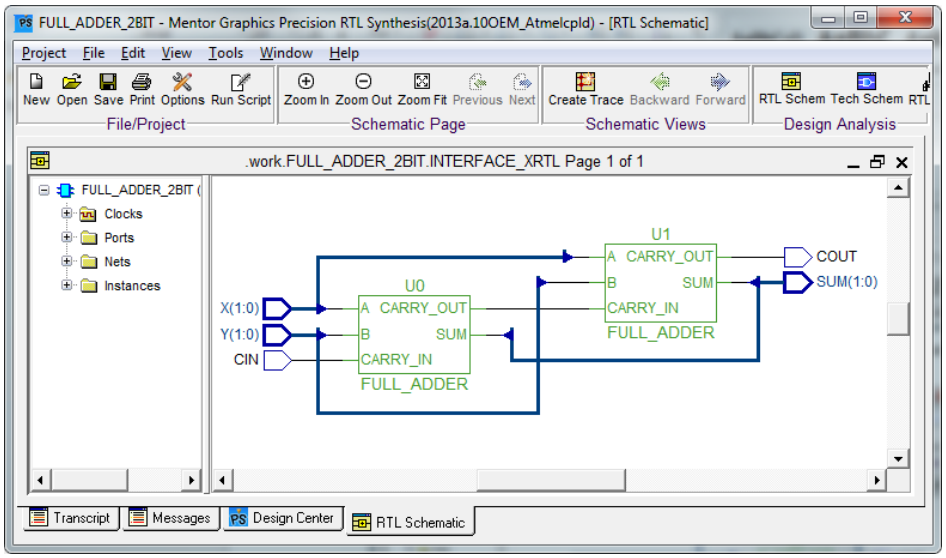
10. Uncheck ***Run Precision in shell mode*** and ***Exit tool when compiled*** option boxes and then click on the ***Compile*** button to launch the Precision Sythesis in GUI mode.



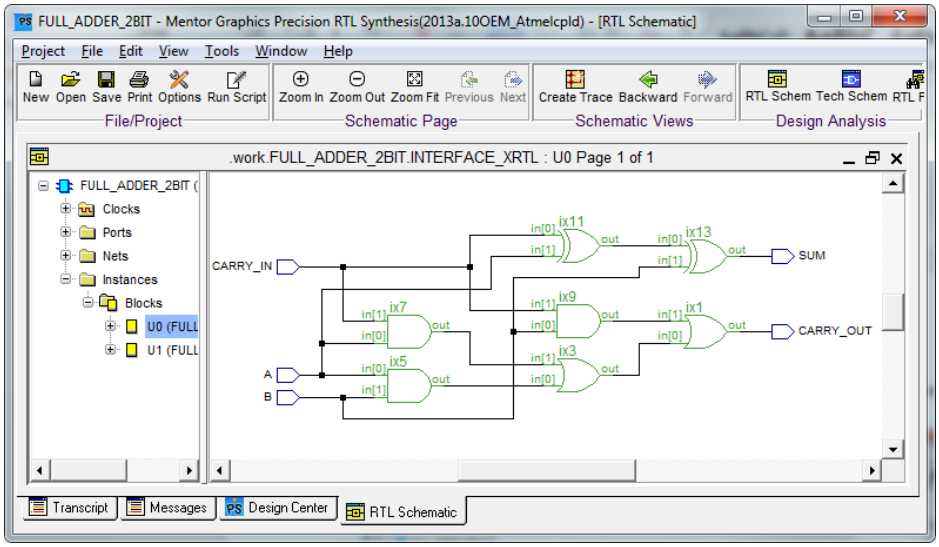
11. Wait for the Precision Synthesis window to launch. Go to ***Project ➔ Open Project***, select ***FULL_ADDER_2BIT.psp*** and then press the ***Open*** button to open this design project in Precision Synthesis.



12. Double-click on ***RTL Schematic*** under **Output Files** in the **Design Center** tab window to view the Register-Transistor level diagram of the top-level design as shown below.



13. Double-click on ***FULL_ADDER*** block in the top-level design RTL schematic to review the sub-level design logic schematic, which is the full adder described in **FULL_ADDER.v**.



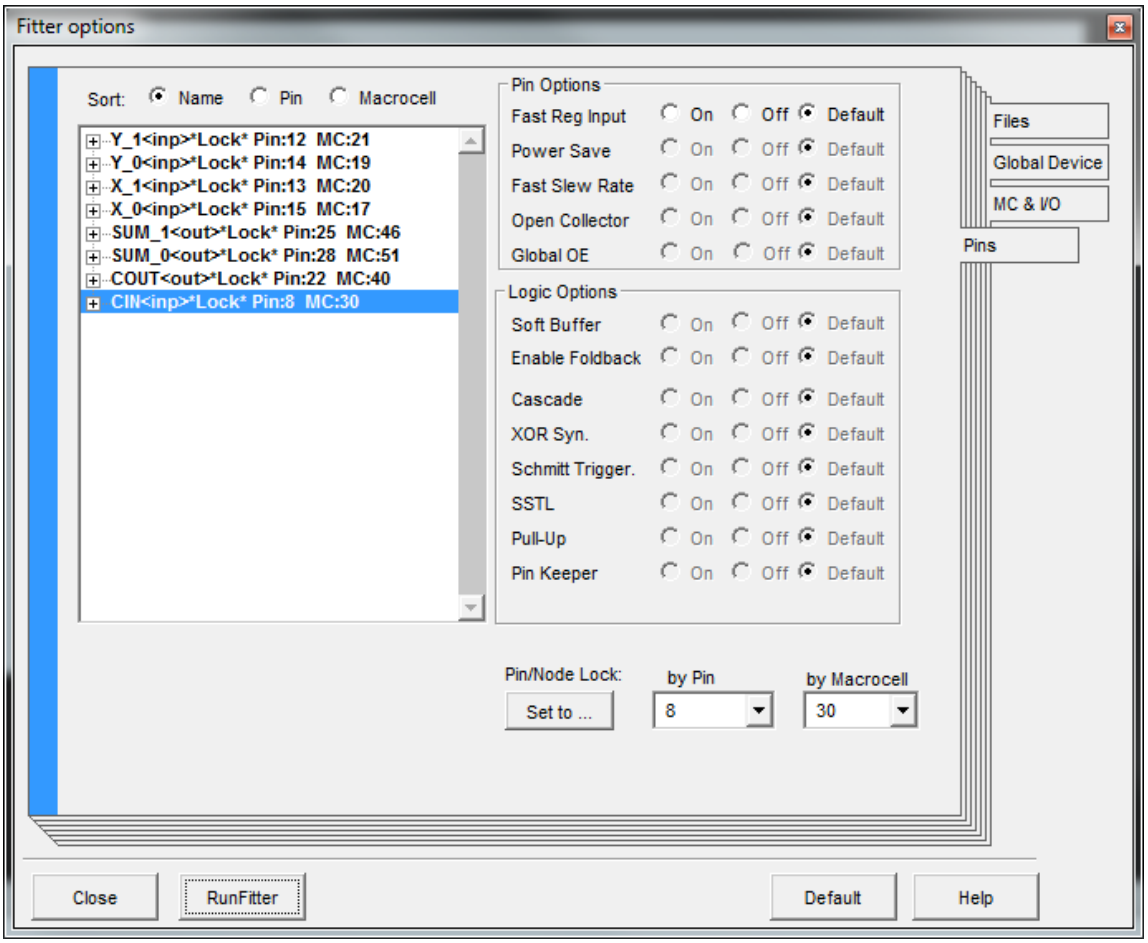
14. In Precision Synthesis, select ***File*** ➔ ***Exit*** and then select ***Exit Without Saving*** button to close Precision Synthesis.

15. In the Log file window, select ***File*** ➔ ***Exit*** to close the log file.

Device Fitting Using Atmel Fitter

1. Press the ***Atmel Fitter*** button under **Device Fitter** of the **Design Flow** tab window to open the **Fitter Options** window.

2. In the **Fitter Options** window, go to ***Global Device*** tab and enable/check the ***Keep*** option for **Pin Fit Control** to keep the pre-assigned and locked input and I/O pin assignments. Pin assignments will be performed in the steps below.
3. Go to the ***Pins*** tab to see the input and output signals.
4. Highlight ***Y_1 <inp>*** in the window on the left and then select ***12*** under the **by Pin** drop-down menu to lock this signal to a push-button switch on the ATF15xx-DK3 board.
5. Highlight ***Y_0 <inp>*** in the window on the left and then select ***14*** under the **by Pin** drop-down menu to lock this signal to a push-button switch on the ATF15xx-DK3 board.
6. Highlight X_***1 <inp>*** in the window on the left and then select ***13*** under the **by Pin** drop-down menu to lock this signal to a push-button switch on the ATF15xx-DK3 board.
7. Highlight ***X_0 <inp>*** in the window on the left and then select ***15*** under the **by Pin** drop-down menu to lock this signal to a push-button switch on the ATF15xx-DK3 board.
8. Highlight ***SUM_1 <out>*** in the window on the left and then select ***25*** under the **by Pin** drop-down menu to lock this signal to a LED on the ATF15xx-DK3 board.
9. Highlight ***SUM_0 <out>*** in the window on the left and then select ***28*** under the **by Pin** drop-down menu to lock this signal to a LED on the ATF15xx-DK3 board.
10. Highlight ***COUT <out>*** in the window on the left and then select ***22*** under the **by Pin** drop-down menu to lock this signal to a LED on the ATF15xx-DK3 board.
11. Highlight ***CIN <inp>*** in the window on the left and then select ***8*** under the **by Pin** drop-down menu to lock this signal to a push-button switch on the ATF15xx-DK3 board.



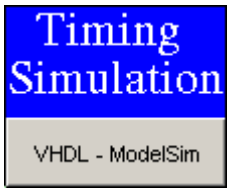
12. Press the ***RunFitter*** button to fit the design and to generate a JEDEC programming file (.JED) for the selected device type. The Fitter Report file (.FIT) as well as the *.VO and *.SDO back-annotated simulation files will also be generated at the same time.

13. If there is no fitting issue, the following window will appear. Click ***OK*** to close it.

14. Click on the ***Close*** button to close the **Fitter Options** window. Now, you can review the **Fitter Report** file (**FULL_ADDER_2BIT.fit**) for details of the fitted design. Go to ***File*** ➔ ***Exit*** to close Fitter Report window when done.

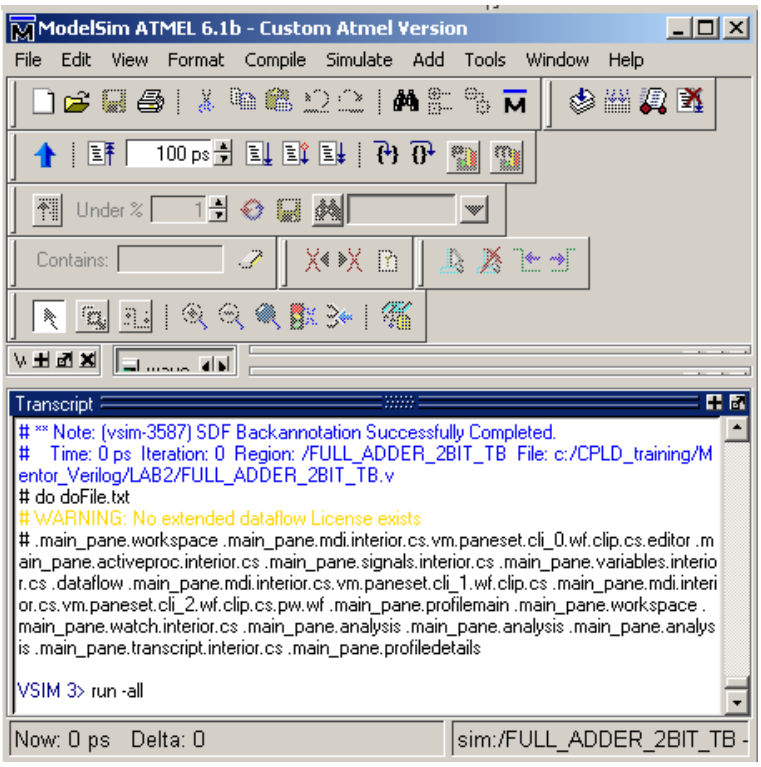
Timing Simulation using ModelSim

1. Since the Verilog testbench file was previously created for functional simulation and added to the Testbench Manager, you can now press the ***Verilog - ModelSim*** button under **Timing Simulation** of the **Design Flow** tab to run timing simulation on the FULL_ADDER_2BIT Verilog design.



2. Highlight ***FULL_ADDER_2BIT_TB.v*** in the **Timing Simulation** window and then click on the ***Simulate*** button to open ModelSim for timing simulation.

3. Wait for ModelSim to launch. Then enter the ***run –all*** command in the **Transcript** window of ModelSim to run the entire simulation time specified in the testbench file.

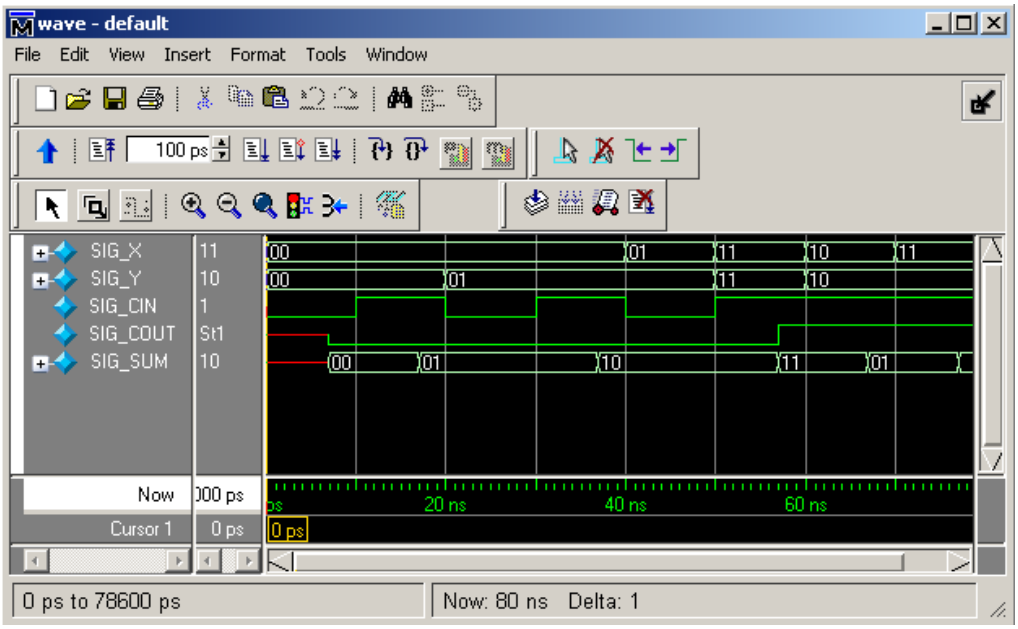


4. In the main ModelSim window, go to ***Simulate*** ➔ ***Break*** to stop the simulation.

5. Use the ***Unlock*** button in the **Wave –default** window to maximize the **Wave** window.

6. Select ***View*** ➔ ***Zoom*** ➔ ***Zoom Full*** in the **Wave** window, select ***Edit*** ➔ ***Select All***, and then select ***Edit*** ➔ ***Delete*** to remove all signals from the waveform.

7. In the **Wave** window, select ***File*** ➔***Load…*** and then select the "**wave.do**" waveform file that was saved during functional simulation and then press ***Open***.



8. Select ***File*** ➔ ***Quit*** in the **Wave** window and then select ***Yes*** to end simulation and close ModelSim.

In-System Programming using ATF15xx-DK3-U and ATMISP

Hardware Step up:

1. Connect the **USB cable** contained in the **ATDH1150USB** kit to the **USB port** of the computer and the **USB connector** of the **ATDH1150USB** cable.

2. Connect the **10-wire ribbon cable** contained in the kit to the **JTAG-A** port of the **ATDH1150USB** and to the JTAG header (**JTAG-IN**) of the **ATF15xx-DK3** board.

   Note:
   The selection jumper at **JP-TDO** on the **ATF15xx-DK3** board should be plugged into the "**TO ISP CABLE**" side.

3. Set both the **VccIO** and **VccINT** selection jumpers on the ATF15xx-DK3 board to ***5V*** if an **ATF15xxAS/ASL** device is being used. If an **ATF15xxASV/ASVL** device is being used, set the **VccIO** and **VccINT** selection jumpers on the ATF15xx-DK3 board to ***3.3V***.

4. Set both the **VccIO** and **VccINT** selection jumpers on the ATF15xx-DK3 board to ***5V***.

   Note:
   The **Power Switch** for the **ATF15xx-DK3** board must be turned ***OFF*** before changing the positions of the **VccIO** and **VccINT** selection jumpers.

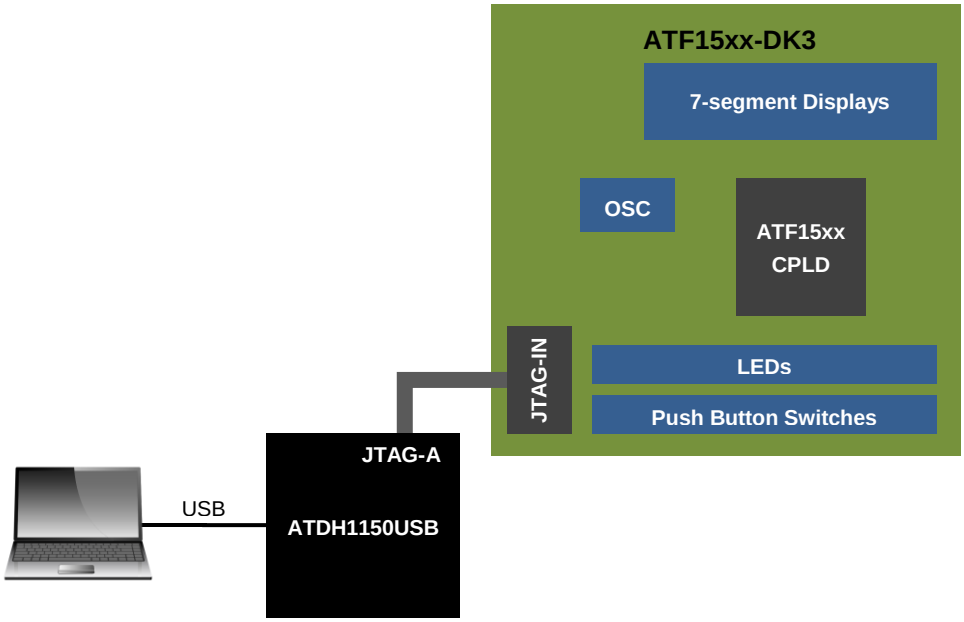
5. Insert a blank **ATF1504AS-10AU44** (or the device type selected in the ProChip Designer project) into the 44-pin TQFP socket. Please note that pin 1 of the device should be at the upper left hand corner of the socket facing the U1 label.

6. Setup the following jumpers on the ATF15xx-DK3 board:

   JPL1 = Set = LED1 = SUM[0];
   JPL2 = Set = LED2 = SUM[1];
   JPL3 = Set = LED3 = COUT;
   JPS1 = Set = Push-button Switch SW1 = X[0];
   JPS2 = Set = Push-button Switch SW2 = Y[0];
   JPS3 = Set = Push-button Switch SW3 = X[1];
   JPS4 = Set = Push-button Switch SW4 = Y[1];
   JPS5 = Set = Push-button Switch SW5 = CIN;

7. Connect the **9V DC** power source to the power connector at **JPower** of the **ATF15xx-DK3** board.

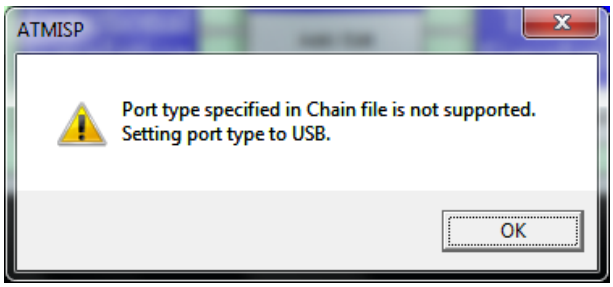
8. Turn the power on by bringing the **Power Switch** of the **ATF15xx-DK3** board to the ***ON*** position.

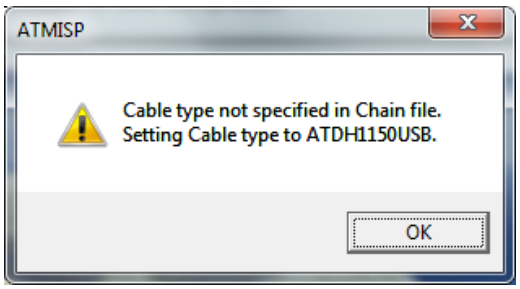


Software Setup:

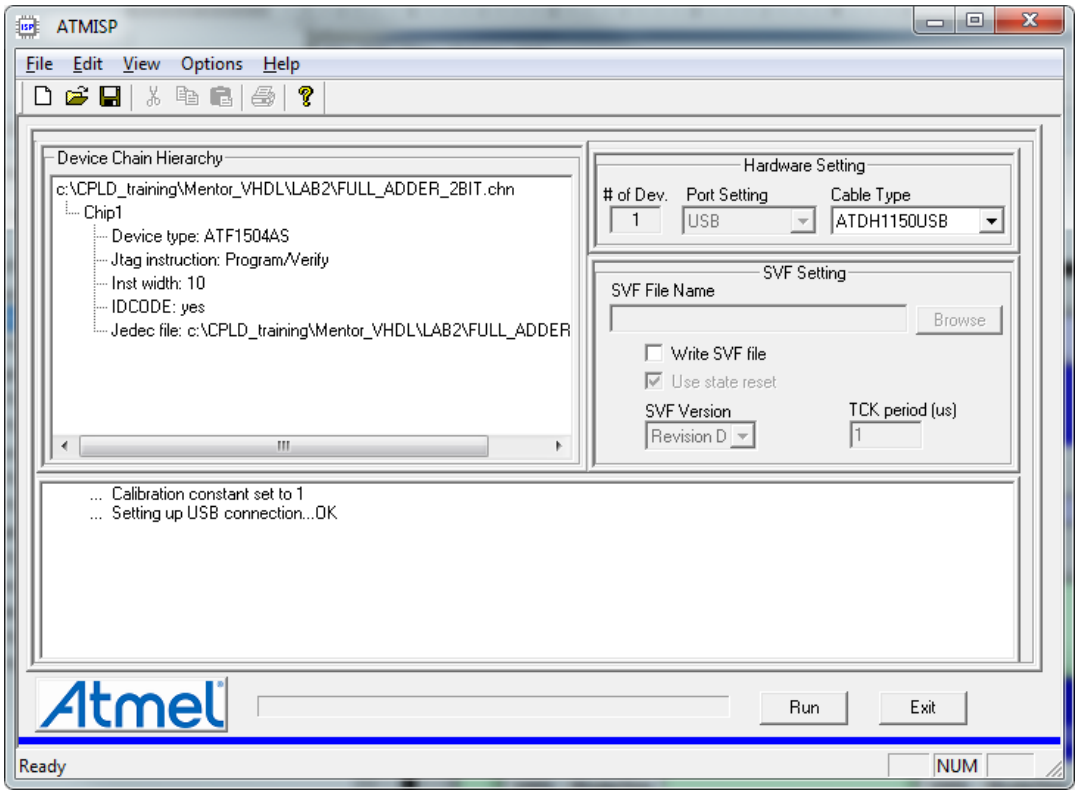
1. Press the ***Program Chip*** button under **Atmel-ISP** of the **Design Flow** tab window.

2. Click ***OK*** on the two **ATMISP** warning message prompts about the port and cable types that pop up when ATMISP is being launched. ATMISP v7.x only supports the USB port type and ATDH1150USB cable type.





3. ATMISP opens and automatically loads the Chain file (.CHN) created by ProChip Designer for the current design.



4. Press the ***Run*** button in the **ATMISP** window to start the JTAG In-system programming.
5. When programming is successfully done, press ***OK*** and then select **File ➔ Exit** to close ATMISP software.
6. Select ***Project ➔ Save***, and then ***Project ➔ Exit*** in Prochip Designer to save the design and then exit the software.

Note:
If ATMISP is not opened from ProChip Designer, but it is opened in stand-alone mode, the following procedure must be setup manually by user in order to perform JTAG ISP.

Run ATMISP Independent of ProChip Designer:

1. In Windows, go to ***Start ➔ All Programs ➔ ProChip Designer 5.0 ➔ ATMISP*** and click on the ***ATMISP*** icon to launch ATMISP.
2. Select **File ➔ New** in ATMISP and then press ***OK*** to confirm that **1** device is used.
3. Select ***ATF1504AS*** (or the appropriate target device type) as the **Device Name**, ***Program/Verify*** as **JTAG Instruction** in **Device Property** window, and then press the ***Browse*** button.
4. Select **"FULL_ADDER_2BIT.jed"** file from the current design directory, press ***Open*** in the **Browse JEDEC File** window, and then press ***OK*** in the **Device Property** window.
5. Press the ***Run*** button in the **ATMISP** window to start the JTAG In-system programming.
6. When programming is successfully done, press ***OK*** and then select **File ➔ Exit** to close ATMISP software.

Testing Design on ATF15xx-DK3 Board

1. Press and hold the push-button switches for **X_0, X_1, Y_0, Y_1** and **CIN** and check the states of the LEDs on the ATF15xx-DK3 board for **SUM_0, SUM_1** and **COUT** to see if it is behaving as a 2-bit full adder or not.

   LED1 = SUM_0;
   LED2 = SUM_1;
   LED3 = COUT;
   SW1 = X_0;
   SW2 = Y_0;
   SW3 = X_1;
   SW4 = Y_1;
   SW5 = CIN;

   Example:

```
            1        Press and hold SW5 (CIN)
        0   0
  +     1   1        Press and hold SW2 (Y_0) and SW4 (Y_1)
---------------
    1   0   0        LED3 (COUT) = on, LED2 (SUM_1) = off, LED1 (SUM_0) = off
```

END OF TUTORIAL 2

Exercise 1: Implemenation of a 1-bit Comparator Design

Use Tutorial 1 as a reference to implement a 1-bit Comparator using Mentor Graphics' Verilog flow, and then show the result using the push-button switches and LED on ATF15xx-DK3 board.

Hint:
Logic equation for a simple 1-bit Comparator is:

```
assign Q = ~(A ^ B);
```

Exercise 2: Implemenation of a Scrolling Display Design

Use Tutorial 2 as a reference to implement the following multi-level Scrolling Display design using Mentor Graphics' Verilog flow, and then show the result on the ATF15xx-DK3 board.

Note:
The "**Update pin Assignment after each compilation**" option must be enabled before running Logic Synthesis if design code contains pin assignments.

```
/* Verilog Design Code for Scrolling Display design:  f02_44TQFP.v*/

/*  Top-level Design:  f02_44TQFP.v
 *  Company:           Atmel Corporation
 *  Contact:           pld@atmel.com
 *  Date:              1/1/2006
 *  Target Device:     ATF15xx (44-TQFP)
 *  Description:       Scrolling Display Design for ATF15xx-DK3 kit
 *  Design files:      Top-level file:  f02_44TQFP.v;
                       Sub-level file:  CLK_DIVIDER.v
*/

module f02_44TQFP(GCLK1, GCLK2, GCLR, SW_4, SW_3, SW_2, SW_1,
                  DSP1_5, DSP1_4, DSP1_3, DSP1_2, DSP1_1, DSP1_0,
                  DSP4_5, DSP4_4, DSP4_3, DSP4_2, DSP4_1, DSP4_0,
                  LED_4, LED_3, LED_2, LED_1, DOT1, DOT4);
input GCLK1;
input GCLK2;
input GCLR;
input SW_4, SW_3, SW_2, SW_1;
inout DSP1_5, DSP1_4, DSP1_3, DSP1_2, DSP1_1, DSP1_0;
inout DSP4_5, DSP4_4, DSP4_3, DSP4_2, DSP4_1, DSP4_0;
output LED_4, LED_3, LED_2, LED_1;
output DOT1;
output DOT4;
reg  [5:0] DSP1 = 6'b000000;
reg  [5:0] DSP4 = 6'b000000;
wire  [4:1] LED;
wire  [4:1] SW;
wire  CLOCK_OUT;

// Vector pin-lock is not supported in Verilog for Precision
// synthesizer.
// However, individual pin lock is possible.
// The following shows the pin-lock statements in Verilog
// for Precision only.
//pragma attribute GCLK1 pin_number 37
```

```
//pragma attribute GCLK2 pin_number 40
//pragma attribute GCLR pin_number 39
//pragma attribute SW_1 pin_number 15
//pragma attribute SW_2 pin_number 14
//pragma attribute SW_3 pin_number 13
//pragma attribute SW_4 pin_number 12
//pragma attribute DSP1_0 pin_number 27
//pragma attribute DSP1_1 pin_number 33
//pragma attribute DSP1_2 pin_number 30
//pragma attribute DSP1_3 pin_number 21
//pragma attribute DSP1_4 pin_number 18
//pragma attribute DSP1_5 pin_number 23
//pragma attribute DSP4_0 pin_number 3
//pragma attribute DSP4_1 pin_number 10
//pragma attribute DSP4_2 pin_number 6
//pragma attribute DSP4_3 pin_number 43
//pragma attribute DSP4_4 pin_number 35
//pragma attribute DSP4_5 pin_number 42
//pragma attribute LED_1 pin_number 28
//pragma attribute LED_2 pin_number 25
//pragma attribute LED_3 pin_number 22
//pragma attribute LED_4 pin_number 19
//pragma attribute DOT1 pin_number 31
//pragma attribute DOT4 pin_number 11

assign {LED_4, LED_3, LED_2, LED_1} = LED;
assign SW = {SW_4, SW_3, SW_2, SW_1} ;

CLK_DIVIDER U1( .GCLK1(GCLK1),  .GCLK2(GCLK2), .GCLR(GCLR),
                .CLK_OUT(CLOCK_OUT));

            assign DOT1 = CLOCK_OUT;
            assign DOT4 = ~ CLOCK_OUT;
            assign LED[1] = SW[1];
            assign LED[2] = SW[2];
            assign LED[3] = SW[3];
            assign LED[4] = SW[4];

always@ (posedge CLOCK_OUT or posedge GCLR)
begin
        if (GCLR)
        begin
            DSP1 <= 6'b000000;
            DSP4 <= 6'b000000;
        end
        else
        begin
            DSP1[0] <= ~ DSP1[5];
            DSP1[1] <= DSP1[0];
            DSP1[2] <= DSP1[1];
            DSP1[3] <= DSP1[2];
            DSP1[4] <= DSP1[3];
            DSP1[5] <= DSP1[4];

            DSP4[0] <= ~ DSP4[5];
            DSP4[1] <= DSP4[0];
            DSP4[2] <= DSP4[1];
```

```
            DSP4[3] <= DSP4[2];
            DSP4[4] <= DSP4[3];
            DSP4[5] <= DSP4[4];

        end
end

assign {DSP1_5, DSP1_4, DSP1_3, DSP1_2, DSP1_1, DSP1_0} = DSP1;
assign {DSP4_5, DSP4_4, DSP4_3, DSP4_2, DSP4_1, DSP4_0} = DSP4;

endmodule
```

```
/* Sub-level design for scrolling design: CLK_DIVIDER.v */

/***************************/
/*  Sublevel Design code  */
/***************************/

/*    Sub-level Design :   CLK_DIVIDER.v
 *    Company:             Atmel Corporation
 *    Contact:             pld@atmel.com
 *    Date:                1/1/2006
 *    Target Device:       ATF15xx (44-TQFP)
 *    Description:         Scrolling Display Design for ATF15xx-DK3 kit.
 *    Design files:        Top-level file:  f02_44TQFP.v;
                           Sub-level file:  CLK_DIVIDER.v
*/

module CLK_DIVIDER (GCLK1, GCLK2, GCLR, CLK_OUT);
input GCLK1;                // 2MHz clock (positive edge)
input GCLK2;                // Active high Register reset
input GCLR;                 // clock output to drive 7 segment display.
output CLK_OUT;

reg [3:0]    CNT1, CNT2, CNT3, CNT4 = 4'b0000;
wire        iCLK;

assign iCLK = GCLK1 | GCLK2;

always@(posedge iCLK or posedge GCLR)
        if (GCLR)
            CNT1 <= 4'b0000;
        else
            CNT1 <= CNT1 + 1;

always@ (posedge CNT1[3] or posedge GCLR)
        if (GCLR)
            CNT2 <= 4'b0000;
        else
            CNT2 <= CNT2 + 1;

always@ (posedge CNT2[3] or posedge GCLR)
        if (GCLR)
            CNT3 <= 4'b0000;
        else
            CNT3 <= CNT3 + 1;
```

```
always@ (posedge CNT3[2] or posedge GCLR)
        if (GCLR)
            CNT4 <= 4'b0000;
        else
            CNT4 <= CNT4 + 1;

assign  CLK_OUT = CNT4[2];

endmodule
```

```
/* Testbench file for top-level Scrolling Display design: */

/*    Top-level Design :  f02_44TQFP_TB.v
 *    Company:            Atmel Corporation
 *    Contact:            pld@atmel.com
 *    Date:               1/1/2006
 *    Target Device:      ATF15xx (44-TQFP)
 *    Description:        Scrolling Display Design testbench file for
 *                        ATF15xx-DK3 kit.
 *    Design files:       Top-level file:  f02_44TQFP.v;
                          Sub-level file:  CLK_DIVIDER.v
*/

`timescale 1ns/100ps

module f02_44FQFP_TB;

reg   SIG_GCLK1;
reg   SIG_GCLK2;
reg   SIG_GCLR;
reg   SIG_SW_4, SIG_SW_3, SIG_SW_2, SIG_SW_1;
wire  SIG_DSP1_5, SIG_DSP1_4, SIG_DSP1_3, SIG_DSP1_2, SIG_DSP1_1,
      SIG_DSP1_0;
wire  SIG_DSP4_5, SIG_DSP4_4, SIG_DSP4_3, SIG_DSP4_2,
      SIG_DSP4_1, SIG_DSP4_0;
wire  SIG_LED_4, SIG_LED_3, SIG_LED_2, SIG_LED_1;
wire  SIG_DOT1;
wire  SIG_DOT4;


f02_44TQFP U1(.GCLK1(SIG_GCLK1),  .GCLK2(SIG_GCLK2),  .GCLR(SIG_GCLR),
              .SW_4(SIG_SW_4), .SW_3(SIG_SW_3),  .SW_2(SIG_SW_2),
              .SW_1(SIG_SW_1), .DSP1_5(SIG_DSP1_5), .DSP1_4(SIG_DSP1_4),
              .DSP1_3(SIG_DSP1_3),  .DSP1_2(SIG_DSP1_2),
              .DSP1_1(SIG_DSP1_1),  .DSP1_0(SIG_DSP1_0),
              .DSP4_5(SIG_DSP4_5),  .DSP4_4(SIG_DSP4_4),
              .DSP4_3(SIG_DSP4_3),  .DSP4_2(SIG_DSP4_2),
              .DSP4_1(SIG_DSP4_1),  .DSP4_0(SIG_DSP4_0),
              .LED_4(SIG_LED_4),   .LED_3(SIG_LED_3),
              .LED_2(SIG_LED_2),   .LED_1 (SIG_LED_1),
              .DOT1(SIG_DOT1),  .DOT4(SIG_DOT4));

// 2MHz is 250 ns.
initial
begin
    SIG_GCLK1 = 1'b0;
    SIG_GCLK2 = 1'b0;
```

```
end

always
begin
#250 SIG_GCLK1  = ~ SIG_GCLK1;  SIG_GCLK2  = ~ SIG_GCLK2;
end

initial
begin

#0       SIG_GCLR <= 1'b1;           //  Global reset
#1000    SIG_GCLR <= 1'b0;

//  After reset, start to show the value from the 1st and 4th 7 segment //
displays
                                 //   Must holding the GCLR push button.
         SIG_SW_4 <= 1'b1;
         SIG_SW_3 <= 1'b1;
         SIG_SW_2 <= 1'b1;
         SIG_SW_1 <= 1'b1;      // Turn on LED1, LED2, LED3, and LED4.
#100000 ;
end

endmodule
```

END OF EXERCISE 2