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 [AND2.v]

Tutorial 2: 2-bit Full Adder Design [FULL_ADDER_2BIT.v]

Exercise 1: Implementation of a 1-bit Comparator

Exercise 2: Implementation of a Scrolling Display Design



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.

Source Manager	
Project Directory:	g\Mentor_Verilog\LAB1
	New Design File
New	Filename: AND_2.v
Edit	Accept Cancel Help
Add	
Remove	▼
Close	Add as Top-level Design

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.

New module wizard 8 de for AND-OR-INVERT library IEEE; e IEEE.STD LOGIC 1164.ell IDLPlanner Module: AND_2 A, B, C, D: in STD LOGIC; F : out STD LOGIC n Port name: Mode: Type: scalar output • • Add Port * × 0 0 Remove Port Ports: input A input B output Q Create Cancel

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;
A HDLPlanner: AN
<u>F</u> ile <u>E</u> dit Ve <u>r</u> ilog
😼 🗅 🛥 🖬 🎒 👗 🖻 1
Text: Medium
module AND_2 (A,B,Q) ;
input A;
input B;
//begin
assign Q = A & B;
endmodule 🗸
File: c:\CPLD_trainin Ln 8 EDI

- 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.





 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.

Testbench Manag	jer 📧
Project Directory	y: ng\Mentor_Verilog\LAB1
New Edit Add	New Test File Filename: AND2_TB.v Accept Cancel
Remove	
Close	Help

- 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.

Functional Simulation	×
Project Directory:	
c:\CPLD_trainingWentor_Verilog\LAB1	
File/Entity	
■-AND2_TB.v	
Close Simulate	Help



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 toplevel 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 **Transcrip**t 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 **I** 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.



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	E Logic Sy	ynthesis wir	ndow to run	Precision Syn	thesis.
----	--------------	-----------------------	------------	--------------	-------------	---------------	---------

Logic Synthesis		
Tool		
Verilog - Precisi	on (Mentor Graphics)	
- Project Directory -		
c:\CPLD_training	Mentor_Verilog\LAB1	
Input		
Top Level Design	File AND_2.v	
Output		
Edif File	AND_2.edf	
	View file when compiled	
Log File	AND_2.log	
	Vew me when complied	
Update Pin/Node	Signal Names and Pin Assignments after each Compilation	
Exit tool whe	n compiled	
Close Compile		Default Help

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.

	C:\ATM	EL_PLS_Tools T1D\Precision\bin\precision.exe	
H	Info:	[40000]: Last compiled on Aug 7 2013 12:00:04	
H	Info:	1445121: Initializing	
	Info.	1445221. Noot Nodule HND_2. rre-processing	
	Info:	[44842]: Compilation successfully completed	
ł	Info:	[44866]: Total lines of BTL compiled: 7.	
	Info:	[44835]: Total CPU time for compilation: 0.3 secs.	
	Info:	[44513]: Overall running time for compilation: 2.0 secs.	-
Į.	Info:	[654]: Current working directory: C:/CPLD_training/Mentor_Verilog/LAB1/A	
Þ	ID_2.		
	Info:	[15329]: Doing rtl optimizations.	
	Info:	[657]: Finished compiling design.	
	Info:	[654]: Current working directory: C:/CPLD_training/Mentor_Verilog/LAB1/A	
	D_2		
H	Into	Uptimizing netlist .work.HND_2.INIERFHGE	
H	Into	Final Design Rule Check.	
H	Info:	198351: KUNNING TIMING OPTIMIZATION FOF design .WOFK.HND_2.INTERFHCE.	
H	Info:	19932]: Starting DNC check	
H	Info.	199331. Linu bag file: C:/CPID twaining/Menton Henilog/IAP1/AND 2/AND 2 a	
	f Into-	10271. Writing file. 0.7011D_training/lientor_verilog/indi/nnD_2/nnD_2.e	
ł	Info:	[657]: Finished sunthesizing design.	
H	Info:	[11019]: Total CPII time for sunthesis: 0.0 s secs.	
	Info:	[11020]: Overall running time for synthesis: 0.1 s secs.	
			r



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.

Atmel Text Viewer: AND_2.log	- O X
<u>F</u> ile <u>V</u> iew <u>E</u> dit <u>H</u> elp	
🚯 🖬 🚭 🐰 🖻 🛍 🛤 春 🛨 → 🚧 Default (Fixed)	<u>-</u>
# Info:	^
# Info: ************************************	
# Info: Library: work Cell: AND_2 View: INTERFACE # Info: ************************************	
# Info: Cell Library References Total Area	
# Info: AND2 atfl5as I x I I AND2 # Info: BUF atfl5as 1 x 1 1 BUF	
# Info: INBUF atf15as 2 x 1 2 INBUF # Info: Number of ports: 3	
# Info: Number of nets : 6	
# Info: Number of instances : 4 # Info: Number of references to this view : 0	
# Info: Total accumulated area :	
# Info: Number of BUF : 1	
# Info: Number of INBUF : 2 # Info: Number of accumulated instances : 4	
# Warning: [9526]: Discarded unsaved work in implementation	AND_2.
<pre># Info: [9530]: Closed project: C:/CPLD_training/Mentor_Verilog/LAB1/AND_2.psp.</pre>	
File: c:\CPLD_training\Mentor_Verilog\LAB1\AND_2.log	n 123 Col 49

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.

E AND_2.log AND_2.log AND_2.log

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.

Fitter options		
Tool		Files
c:\ATMEL_P	LS_Tools T1D\Prochip\\PLDFit\fit1504.exe	Global Device
Project Direct	ory	MC & VO
c:\CPLD_trai	ining\Mentor_Verilog\LAB1	Pins
Edif File	AND_2.edf	
Output Jedec File	AND_2.jed	
Report File	NNU_2.int √ view file after fitting	
UES (2 ASCII Cha	racters)	
Close	RunFitter Default	Help

3. Select the *Global Device* tab.

Global Configuration	Power Reset C Large Hysteresis Small Hysteresis	Pin Fit Control		Files Global Device
Power Save Pin Power Down 1 Pin Power Down 2 GCLK Auto Wake C GCLK1 Auto Wake C GCLK2 Auto Wake C GCLK3 Auto Wake	Device Logic Options	if necessary		Pins
Generate Sim Files	Security			
Close RunFitter			Default	Help

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.
- 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.

Fitter options					
		h			
Sort: Name Pin Macrocell	Pin Options				
n. Ocoutbill ookt Din:28. MC:54	Fast Reg Input	C On C Off • Default Files			
B <inp>*Lock* Pin:15 MC:17</inp>	Power Save	C On C Off C Default Global Device			
A <inp>*Lock* Pin:14 MC:19</inp>	Fast Slew Rate	○ On ○ Off ● Default			
	Open Collector	○ On ○ Off Default			
	Global OE	C On C Off C Default Pins			
	Logic Options				
	Soft Buffer	C On C Off			
	Enable Foldback	C On C Off 🖲 Default			
	Cascade	○ On ○ Off Default			
	XOR Syn.	○ On ○ Off Default			
	Schmitt Trigger.	C On C Off C Default			
	SSTL	C On C Off C Default			
	Pull-Up	C On 🖲 Off C Default			
	Pin Keeper	C On C Off C Default			
	-				
·					
	Pin/Node Lock:	by Pin by Macrocell			
	Set to	T			
Close RunFitter		Default Help			

- 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.

And and	
Design successfully fit.	

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

 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.

Timing Simulation	X
Project Directory:	
c:\CPLD_training\Mentor_Verilog\LAB1	
File/Entity/Component	
	A
	-1
C Max C Min	
Close Simulate	Help
	1040



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

ModelSim ATMEL 6.1b - Custom Atmel Versio	on _ 🗖 🗶
File Edit View Format Compile Simulate Add	Tools Window Help
📙 🗅 🞜 🗶 🖕 🐇 🐚 🛍 😂 😂 🖌 🖄	: 😘 🖬 🛛 🍪 🕮 🚑 🕅 👌 🚹 🔠 🗆 100 ps 🗄 💷 🚉 ?) 70 🕎 🌇 🌇
1 🕈 🖓 🖬 🕅	🔽 🛛 Contains: 🔽 🖉 🕅 🗶 🗶 🖿 🖉
Watch # # # # # AND2_T	
Transcript -	
# vsim-L cpld_ver -do doFile.txt -sdftyp /AND2_TB/U1 # Loading work.AND2_TB # Loading c:VATMEL_PLStools\ModelSim61bADEM\m # tooking c:VATMEL_PLStools\ModelSim61bADEM\m # main_company tooking c:VATMEL_PLStools\ModelSim61bADEM\m # tooking c:VATMEL_PLStools\ModelSim61bADEM\m # Time: 0 ps Iteration: 0 Region: /AND2_TB File: c # do doFile.txt # wARNING: No extended dataflow License exists # .main_pane.workspace .main_pane.mdi.interior.cs.vn _pane.variables.interior.cs .dataflow .main_pane.mdi.interior.cs.vn _pane.variables.interior.cs .dataflow .main_pane.mdi.interior.cs.vn _pane.variables.interior.cs .main_pane.modi.interior.cs.vn _vanu_pane.profile.txt _vanu_pane.profile.txt _vanu_pane.profile.txt _vanu_pane.profile.txt _vanu_pane.profile.txt _vanu_pane.profile.txt _vanu_pane.profile.txt _vanu_pane.profile.txt _vanu_pane.profile.txt _vanu_pane.profile.txt _vanu_pane.profile.txt _vanu	L=c:/CPLD_training/Mentor_Verilog/LAB1/AND2.sdo AND2_TB modeltech\win32aoem//atmel/verilog/cpld_ver.TRI modeltech\win32aoem//atmel/verilog/cpld_ver.BUF modeltech\win32aoem//atmel/verilog/cpld_ver.XDR2 modeltech\win32aoem//atmel/verilog/cpld_ver.XDR2 modeltech\win32aoem//atmel/verilog/cpld_ver.AND2 or_Verilog/LAB1/AND2_TB.v(25): [T0FD] - System task or function '\$finished' is not defined. attraction to the term of term o
Now: O ps Delta: O	sim:/AND2_TB - Limited Visibility Region Covergroup Coverage: 0% Recursive M

- 4. In the main ModelSim window, go to *Simulate* → *Break* to stop the simulation.
- 5. Use the **Unlock** button **I** 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.
- 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.

- 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.



 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.

ATMISP	
<u>File E</u> dit <u>V</u> iew Options <u>H</u> elp	
Device Chain Hierarchy c:\CPLD_training\Mentor_Verilog\LAB1\AND_2.chn Chip1 Device type: ATF1504AS Jtag instruction: Program/Verify Inst width: 10 Jedec file: c:\CPLD_training\Mentor_Verilog\LAB1\AND_2.jed	Hardware Setting # of Dev. Port Setting Cable Type 1 USB ATDH1150USB SVF File Name Write SVF file Write SVF file SVF Version TCK period (us) Bevision D 1
Calibration constant set to 1 Setting up USB connectionDK	Run Exit
Ready	

- 4. Press the *Run* button in the **ATMISP** window to start the JTAG In-system programming.
- 17 ProChip Designer v5.0 Mentor Graphics Verilog Design Flow Tutorial CPLD_Mentor_Verilog_tutorial.pdf [01/28/2016]



- 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

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.



New module wizard		
Module: FULL_ADDER		- VHDL code for AND-OR-INVERT g library IEEE; use IEEE.STD_LOGIC.1164.xli; HUDD I: sport (A. B. C. D. in STD_LOGIC; F: out ISTD_LOGIC;);
Port name: Ports:	Mode: output	Type: Scalar Add Port
input A input B input CARRY_IN output SUM output CARRY_OUT		×
		Create Cancel

- 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);
HDLPlanner: FULL_ADDER.v
<u>File E</u> dit Ve <u>r</u> ilog
Text: Medium
module FULL_ADDER (A,B,CARRY_IN,SUM,CARRY_OUT) ;
input A;
input B;
input CARRY_IN;
output SUM;
output CARRY_OUT;
//begin
assign SUM = A ^ B ^ CARRY_IN;
assign CARRY_OUT = (A & B) (A & CARRY_IN) (B & CARRY_IN);
endmodule 🗸
File: c:\CPLD_training\Mentor_Verilog\LAB2\FUL Ln 12 Co EDITING



- 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.
- 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.



- 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;</pre>
#10
SIG X <= 2'b01;
SIG Y <= 2'b01;
SIG CIN <= 1'b0;</pre>
#10
SIG X <= 2'b11;
SIG Y <= 2'b11;
SIG CIN <= 1'b1;</pre>
#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 "**xx**" was intentionally entered to the testbench code.

	Log	
	"c:\CPLD_training\Mentor_Verilog\LAB2\FULL_ADDER_2BIT_TB.v""	
	** Error: c:\CPLD_training\Mentor_Verilog\LAB2\FULL_ADDER_2BIT_TB.v(25): (vlog-2730) Undefined variable: 'XY'.	
ļ	** Error: c:\CPLD_training\Mentor_Verilog\LAB2\FULL_ADDER_2BIT_TB.v(25): near "SIG_Y": syntax error, unexpected IDENTIFIER, expecting ';'	-



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.

```
E·▶ Source Manager : [c:\CPLD_training\Mentor_Verilog\LAB2]

⊡ E FULL_ADDER_28IT.v
```

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.

Functional Simulation	×
Project Directory:	
c:\CPLD_training\Mentor_Verilog\LAB2	
File/Entity	
FULL_ADDER_2BIT_TB.v	<u></u>
	<u></u>
Close Simulate	Help

10. Wait for ModelSim to launch. Then enter the *run –all* command in the **Transcrip**t 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 **I** 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.
- 27 ProChip Designer v5.0 Mentor Graphics Verilog Design Flow Tutorial CPLD_Mentor_Verilog_tutorial.pdf [01/28/2016]



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

Logic Synthesis
Tool Info Verilog - Precision (Mentor Graphics) Info
Input Top Level Design File FULL_ADDER_2BIT.v
Couput Edif File FULL_ADDER_2BIT.edf View file when compiled Log File FULL_ADDER_2BIT.log
view file when compiled Update Pin/Node Signal Names and Pin Assignments after each Compilation Run Precision in shell mode Exit tool when compiled
Close Compile Default Help

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.





- 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.
- 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.

Fitter options	_	
		h
Sort: Name C Pin C Macrocell	Fast Reg Input	C On C Off Default
Y_1 <inp>*Lock* Pin:12 MC:21 ▲</inp>	Power Save	C On C Off C Default
⊕Y_0 <inp>*Lock* Pin:14 MC:19 ⊕X_1<inp>*Lock* Pin:13 MC:20</inp></inp>	Fast Slew Rate	C On C Off C Default
	Open Collector	C On C Off Default MC & VO
SUM_1 <out>*Lock* Pin:25 MC:46</out>	Clobal OF	C On C Off O Default Pins
COUT <out>*Lock* Pin:20 MC:51 ⊡ COUT<out>*Lock* Pin:22 MC:40</out></out>	Giobal OE	
CIN <inp>*Lock* Pin:8 MC:30</inp>	Coff Buffer	C. o. C. Off C. Dofault
	Son Burrer	
	Enable Foldback	
	Cascade	C On C Off • Default
	XOR Syn.	C On C Off © Default
	Schmitt Trigger.	C On C Off (Default
	SSTL	C On C Off C Default
	Pull-Up	C On C Off @ Default
	Pin Keeper	C On C Off C Default
	Pin/Node Lock:	by Pin by Macrocell
	Set to	8 • 30 •
Close RunFitter		Default Help



- 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.



 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

 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 **Transcrip**t 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 **I** 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.

- 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.
- 32 ProChip Designer v5.0 Mentor Graphics Verilog Design Flow Tutorial CPLD_Mentor_Verilog_tutorial.pdf [01/28/2016]



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.



 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.

ATMISP	
<u>File E</u> dit <u>V</u> iew Options <u>H</u> elp	
🗋 🗅 🚅 🔚 X 🖻 💼 🎒 💡	
Cevice Chain Hierarchy C:\CPLD_training\Mentor_VHDL\LAB2\FULL_ADDER_2BIT.chn Device type: ATF1504AS Jtag instruction: Program/Verify Inst width: 10 IDCODE: yes Jedec file: c:\CPLD_training\Mentor_VHDL\LAB2\FULL_ADDER Calibration constant set to 1 Setting up USB connectionOK	Hardware Setting # of Dev. Port Setting Cable Type 1 USB CATDH1150USB SVF File Name Write SVF file Use state reset SVF Version TCK period (us) Revision D 1
Atmel	Run Exit
Ready	NUM

- 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 = LED2 = LED3 = SW1 = SW2 = SW3 = SW4 = SW5 =	: SUI : SUI : COI X_0; Y_0; X_1; Y_1; CIN;	M_0; M_1; UT;	
Examp	le: 0	1 0	Press and hold SW5 (CIN)
+ 1	1 0	1 	Press and hold SW2 (Y_0) and SW4 (Y_1) 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 $Q = \sim (A ^ B);$

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
* Top-level besign: To2_41gill.*
* 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;
Cub level file: GLK_DIVIDEP 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 = \sim 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];</pre>
            DSP1[1] <= DSP1[0];</pre>
            DSP1[2] <= DSP1[1];</pre>
            DSP1[3] <= DSP1[2];</pre>
            DSP1[4] <= DSP1[3];</pre>
            DSP1[5] <= DSP1[4];
            DSP4[0] <= ~ DSP4[5];
            DSP4[1] \le DSP4[0];
            DSP4[2] <= DSP4[1];</pre>
```



```
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
 *
                        pld@atmel.com
     Contact:
 *
                         1/1/2006
    Date:
 *
                       ATF15xx (44-TQFP)
     Target Device:
 *
    Description:
                        Scrolling Display Design for ATF15xx-DK3 kit.
*
                        Top-level file: f02 44TQFP.v;
    Design files:
                         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 \leq CNT2 + 1;
always@ (posedge CNT2[3] or posedge GCLR)
       if (GCLR)
           CNT3 <= 4'b0000;
       else
           CNT3 <= CNT3 + 1;
```

```
38 ProChip Designer v5.0 Mentor Graphics Verilog Design Flow Tutorial
CPLD_Mentor_Verilog_tutorial.pdf [01/28/2016]
```



```
always@ (posedge CNT3[2] or posedge GCLR)
    if (GCLR)
        CNT4 <= 4'b0000;
    else
        CNT4 <= CNT4 + 1;
assign CLK_OUT = CNT4[2];
endmodule</pre>
```

```
/* Testbench file for top-level Scrolling Display design: */
/*
      Top-level Design : f02 44TQFP TB.v
 *
                            Atmel Corporation
      Company:
 *
      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;
req
      SIG GCLK1;
      SIG GCLK2;
req
      SIG GCLR;
rea
      SIG SW 4, SIG SW 3, SIG SW 2, SIG SW 1;
req
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 \overline{SW} 4), .SW 3(SIG \overline{SW} 3), .SW 2(SIG \overline{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;</pre>
         SIG SW 2 <= 1'b1;
         SIG SW 1 <= 1'b1; // Turn on LED1, LED2, LED3, and LED4.
#100000 ;
end
endmodule
```

END OF EXERCISE 2

