





© 2001 by X Engineering Software Systems Corp., Apex, North Carolina 27502

All rights reserved. No part of this text may be reproduced, in any form or by any means, without permission in writing from the publisher.

The author and publisher of this text have used their best efforts in preparing this text. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this text. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

XESS, XS40, and XS95 are trademarks of X Engineering Software Systems Corp. XILINX, Foundation, XC4000, and XC9500 are trademarks of XILINX Corporation. Other product and company names mentioned are trademarks or trade names of their respective companies.

The software described in this text is furnished under a license agreement. The software may be used or copied under terms of the license agreement.

Table of Contents

Table of Contents	i
0	3
Introduction	3
1	6
VHDL-Based Design	6
In this chapter you will learn how to:	6
Overall Design Flow	6
Starting a Project	9
Creating the VHDL Source Code	14
Synthesizing a Netlist	
Running a Simulation	
Implementing the Design	56
Downloading and Testing	72
Retargeting the Circuit at an XC9500 CPLD	75
2	97
Schematic-Based Design	97
In this chapter you will learn how to:	97
Overall Design Flow	97
Building the Counter Circuit for the XS40 Board	
Starting the Project	
Starting the Schematic Editor	100
Adding Components to the Schematic	101
Connecting the Components	107
Adding and Connecting to a Bus	

	Assigning Pins to the Inputs and Outputs	123
	Creating, Checking, and Exporting the Netlist	135
	Running Simulations with the Schematic Editor	139
	Implementing, Downloading, and Testing the Counter Circuit	154
	Examining the Counter circuit with the FPGA Editor	159
B	uilding the Counter Circuit for the XS95 Board	169

2

Schematic-Based Design

In this chapter you will learn how to:

- Use the schematic editor in Foundation 2.1i to create a logic circuit.
- Use the schematic editor as a graphical front-end for the logic simulator.
- Extract the netlist from a schematic for further processing by the rest of the tools in Foundation 2.1i.
- Examine the arrangement of circuit components over the array of resources in the FPGA or CPLD using the FPGA Editor or CPLD ChipViewer tools.

Overall Design Flow

The schematic-based design flow (Figure 1) is almost identical to the VHDL-based design flow of the last chapter. The main difference is that the logic circuit is entered graphically using a schematic editor instead of as VHDL source code using a text editor. The graphical editor extracts the netlist from the drawn circuit and exports it to the rest of the Foundation 2.1i tools for simulation, implementation, and bitstream generation. The latter steps are accomplished the same way as was done in the VHDL-based design flow.

In the rest of this chapter we will follow the design flow of Figure 1 while designing a simple counter circuit. The 24-bit counter will take 50 MHz clock signal from a programmable oscillator on the XS40 or XS95 Board and the top and bottom segments of the LED digit are driven by the upper two bits of the counter. The upper segment will flash approximately three times each second (50,000,000 / 2^{24} = 2.98) and the lower segment will blink twice as fast.



Figure 4: Steps in creating and testing an FPGA or CPLD-based design.

Building the Counter Circuit for the XS40 Board

Starting the Project

Once again we start the Foundation 2.1i software and click on the Create a New Project radio-button to begin our schematic-based design.

Getting Started	×
C Open an Existing Project dsgn1_1 leddcd piiitst ppiii test2 invtest	More projects
c:\prag21i\dsgn1_1	
Create a <u>New Project</u>	
Always open last project	
<u>O</u> K <u>C</u> ancel	<u>H</u> elp

In the **New Project** window that appears, I set the design name to **dsgn2_1** and elected to store the project design files in the C:\PRAG21I directory. Select a schematic-based design flow by clicking the Schematic radio-button if it isn't already selected. Then choose the FPGA or CPLD family, device, and speed grade in the three fields at the bottom of the window. I will target this design at an XS40-005XL Board, so I selected the XC4005XLPC84 FPGA of the XC4000XL FPGA family with a -3 speed grade.

New Proje	ect		×
Name:	dsgn2_1		ОК
Directory:	C:\PRAG211		Cancel
Туре:	E2 1i	-	<u>Browse</u> Help
Flow:	© <u>S</u> chematic		
XC4000X	L 🔄 4005XLPC	:84 💌	3 •

Starting the Schematic Editor

After clicking OK in the **New Project** window, the **Project Manager** window will appear as shown below. Since we are targeting the XC4005XL FPGA, the xc4000x library is automatically added to the file hierarchy of the project. To begin building the design, click on the Schematic Editor icon in the **Project Flow** pane.

🔹 dsgn2_1 - 4005XLPC84-3 - Proje	ect Manager	<u>_ </u>
<u>File Document View Project Impler</u>	mentation <u>T</u> ools <u>H</u> elp	
Files Versions ☐ dsgn2_1 ☐ dsgn2_1 ☐ simprims ☐ xc4000x	Flow Contents Reports Synthesis dsgn2_1 (4005XLPC84-3) DESIGN ENTRY DESIGN ENTRY Schematic Editor Schematic Editor IMPLEMENTATION PROGRAMMING	
Pcm : Opening project: c:)		
Pcm : Design Type Schematic		
	ब	▼
Ready		

Adding Components to the Schematic

An empty **Schematic Editor** window will appear. The schematic is named DSGN2_11.SCH by default, but you can change it later when you save it. We need components to construct a circuit schematic, so click on the Symbols toolbox icon.

抣 Se	chematic Editor - [l	DSGN2_11.SCH]	
<u>E</u>	ile <u>E</u> dit <u>M</u> ode <u>O</u> p	tions <u>H</u> ierarchy <u>V</u> iew <u>D</u> isplay <u>T</u> ools <u>W</u> indow Hel <u>p</u>	_ & ×
睝	🛎 日 🐛 🎒 🛓	※ 🖻 🖻 🖄 Q Q 🖉 이 세종에 🖻 🕎 😼	
			<u> </u>
H			_
La			
Sym			
큭			
늰			
A			
	र		<u>ب</u> ا
	DSGN2_11		
	0.0 , 0.8	Sets/cancels Symbols mode	Select and Drag

The **SC Symbols** window will appear containing a list of all the components that are available to us for constructing a circuit with the XC4005XL FPGA. The information about these components is stored in the xc4000x library that is part of the project. (Each FPGA and CPLD device family has its own specialized library of components that take advantage of the particular features found in each family.) As you scroll through the list, a short description of each component appears at the bottom of the **SC Symbols** window. Stopping on the CC16CE entry shows that it represents a 16-bit counter. This is the largest counter we can find in the list, so leave this entry highlighted.

Schematic Editor - [Modified - DSGN2_11.SCH]	×
	CB2CLED CB2CLE CB4CLE CB4CLE CB4CLE CB4CLE CB4CLE CB4CLE CB8CLE CB8CLE CB8CLE CB8CLE CB8CLE CB8CLE CC16CLE CC1
5.8, 1.6	Symbols

Once the mouse cursor leaves the **SC Symbols** window and re-enters the **Schematic Editor** window, you will see that a symbol for the 16-bit counter is attached to the cursor.



You can place the counter in the schematic by clicking the left mouse button. The symbol will be labeled with the component name and the input and output terminals will be labeled with their names.



We need a 24-bit counter, so go back to the **SC Symbols** window and find the 8-bit counter CC8CE and add it to the schematic as shown below. These two counters will be cascaded to form the complete 24-bit counter.



The counters need to be connected to a clock signal that will make them increment their values. The clock needs to enter the FPGA chip through an input pin and then get on the clock network in the FPGA that connects all the flip-flops in the FPGA. To do this go back to the **SC Symbols** window, find the IPAD component that represents an FPGA input pin, and add it to the schematic. Then add a clock-buffer (BUFG) component that transfers any signal on its input onto the FPGA clock network. Those are all the components we need for the moment, so remove the **SC Symbols** window by double-clicking in its upper-left corner.



Connecting the Components

We now start connecting these components together by clicking on the Draw wires icon. After doing this, the mouse cursor will appear as a pencil.



Position the mouse cursor over the input pad terminal and click the mouse.



A wire now connects the IPAD to the mouse wherever the mouse moves to. To connect the input pad to the clock buffer, position the mouse over the input terminal of the BUFG symbol and click the mouse.

s S	chematic Editor - [Modified - DSGN2_11.SCH]	
E	File Edit Mode Options Hierarchy View Display Tools Window Help	_ 문 ×
1		
븕		_
- VS 		
쁘		
Ð		
4	CC8CE	
4	Q7.0]	
	CE CEO	
A		
듣		
Ŧ		
	CC16CE	
	Q(163)	
		لى
		<u> </u>
		Duran Mines
	2.4 , 3.4	Draw wires

Now a line connects the IPAD and BUFG symbols, symbolizing a wire connecting the input pin to the clock buffer. The mouse cursor is detached from the wire so it can be used to initiate the creation of other wires.

so Se	chematic Editor - [Modified - DSGN2_11.SCH]	
E	ile <u>E</u> dit <u>M</u> ode <u>O</u> ptions <u>H</u> ierarchy <u>V</u> iew <u>D</u> isplay <u>T</u> ools <u>W</u> indow Help	 ×
1		
24		
뺼		
Ð		
크	CCSCE	
-	Q[7:0]	
F	CE CEO	
A		
<u> </u>		
A		
	Λ	
	Q[150]	
	CE CEO	
	e	<u>ت</u>
	DSGN2 11	
	2.5 , 2.8	Draw Wires

The clock signal can be connected to the clock inputs of both counters by drawing wires from the output of the BUFG symbol to the clock input terminals of both counters as shown below.



In order to function as a 24-bit counter, the 8-bit counter should only increment when the 16-bit counter rolls over from 0xFFFF to 0x0000. (Thus the 8-bit counter increments once for every 65,536 clock pulses.) The 16-bit counter has an output (CEO) that goes to a logic 1 only when the counter value is 0xFFFF. Connecting CEO to the clock-enable input (CE) of the 8-bit counter means the 8-bit counter will only react to a clock pulse whenever the 16-bit counter contains the value 65535.



Unlike the 8-bit counter, the 16-bit counter should increment on every clock pulse so its clock-enable input should be driven to a logic 1 at all times. To do this, re-open the **SC Symbols** window and add a VCC symbol to the schematic. Then attach the VCC symbol to the CE input of the 16-bit counter with a wire. Each counter also has a CLR input that clears the counter to zero when the input is driven high. To keep the counters from inadvertently being cleared during the operation of the circuit, add two GND symbols and attach one to each CLR input.



Adding and Connecting to a Bus

Now the two counters are cascaded and their inputs are set appropriately, so we can turn our attention to the counter outputs. We need to connect the upper two bits of the 8-bit counter output bus to LED segments on the XS40 Board. To gain access to these two bits, we begin by clicking on the Draw buses icon.





Next, click on the output bus terminal of the 8-bit counter and draw a bus downward.

Click the right mouse button to bring up a pop-up window that will be used to terminate the bus. Select the Add Bus Label... entry and release the right mouse button.



The **Add Bus Terminal/Label** window will appear. Type the name for the bus in the Name field (we will use CNT for this example) and then click on OK.

Add Bus Terminal/Label	×
Bus Label (e.g. BUS[0:2]):	🔽 Simple Bus
Name: CNT	Range: [7 ★ : 0 ★]
Terminal Marker:	
OK Bus End	<u>Cancel</u> <u>H</u> elp

The bus is now shown with the CNT label. The **CNT** bus is eight bits wide with indices for the individual bus wires running from 7 down to 0.



Now add output pins (symbol OPAD in the **SC Symbols** window) and output buffers (symbol OBUF). Then wire the output terminal of each output buffer to one of the output pins.



Now we need to tap-off the upper two bits of the **CNT** bus and send them to the output pins. Click on the Draw bus taps icon to initiate this step.



Now click on the bus you want to tap into (**CNT**, in this case). The particular wire in the bus that is being tapped into is shown in the status line at the bottom of the **Schematic Editior** window. You can use the up-arrow and down-arrow keys to change the index of the tapped wire.



Once you have set the correct bus wire to tap, click the mouse cursor on the destination terminal where the bus wire should connect. In the example shown below, the **CNT7** bus wire is being connected to the lower output buffer.



Once you click the mouse on the destination terminal, the labeled bus wire is drawn between the bus and the terminal.



The procedure can be repeated to connect the upper output buffer to bus wire CNT6.



Assigning Pins to the Inputs and Outputs

The circuit is now complete, but now we need to assign the inputs and outputs to the correct pins of the FPGA on the XS40 Board as shown in Figure 5.





We begin with assigning the clock input by right-clicking on the IPAD symbol. Select the Symbol Properties... item in the pop-up menu that appears.



The **Symbol Properties** window shows the values for the various attributes of the input pin. The input pad has already been automatically assigned an internal reference name of **\$I4**.

Symbol Prope	erties				×
Name:	IPAD	-	Mo <u>v</u> a	e Name	
<u>R</u> eference:	\$14		<u>M</u> ove F	leference	
Section:	NO SECTIONS	-	Attri	ibutes	
Eootprint:		-	C <u>o</u> m	ments	
<u>T</u> echnology:		-	Pjn Pa	rameters	
Parameters:	,				
<u>N</u> ame:		•	Change	Add	
Description:			Browse		
EXT=IP	AD			Delete	
LEVEL=	XILINX -2.0.0			Mov <u>e</u>	
LIDVEN	-2.0.0			Display All	
				ear Display	
)isplay File	
			Sy	vmbol Editor	
Apply	<u>0</u> K	Cano	el	Help	

We can substitute a more descriptive reference name (**CLK**) for the internal name as shown below. Click on the Apply button to activate the name change.

Symbol Prope	erties			×
Name:	IPAD 🔹	М	o <u>v</u> e Name	
<u>R</u> eference:		<u>M</u> ov	e Reference	
Section:	NO SECTIONS		Attributes	
<u>F</u> ootprint:			2 <u>o</u> mments	
<u>T</u> echnology:	, 	Pin	Parameters	
Parameters:	,			
<u>N</u> ame:	-	Chang	ge <u>A</u> dd	
Description:		Brows	se	
EXT=IP	AD		Delețe	1
LEVEL=	XILINX		Mov <u>e</u>	1
LIBVEN	=2.0.0		Display All	i
			<u>Clear Display</u>	i
			Display File	1
			Symbol Editor	ĺ
Apply	<u></u> KCa	ncel	Help	

Next we need to assign the location of the physical pin on the FPGA package to this IPAD symbol. In the Parameters section of the window, select LOC from the drop-down list attached to the Name field.

Symbol Prope	rties			×
Name:			Moye Name	
Name.	IPAD		MO <u>v</u> e Mallie	
<u>R</u> eference:	CLK		Move Reference	
Section:	NO SECTIONS	-	Attributes	
<u>F</u> ootprint:		-	Comments	
<u>T</u> echnology:		-	Pin Parameters	
Parameters: -	·			
<u>N</u> ame:	LOC	•	Change	Add
Description:			Browse	
EXT=IP/	4.D			Delete
LEVEL=XILINX LIBVEB=2.0.0				Mov <u>e</u>
				Display All
				lear Display
				Display File
			S	ymbol Editor
Apply	<u>0</u> K	Cano	;el	Help

Next set the pin location into the Description field. The clock signal from the oscillator on the XS40 Board enters the FPGA through pin 13, so type P13 in the field. Then click the Add button.

Symbol Properties							
			L.L.				
Name:	IPAD	•	Mo <u>v</u> e Name				
<u>R</u> eference:	CLK		Move Reference				
Section:	NO SECTIONS	*	Attributes				
<u>F</u> ootprint:		-	Comments				
<u>T</u> echnology:		-	Pin Parameters				
<u>Parameters:</u>		_					
<u>N</u> ame:	LOC	-	Change	e <u>A</u> dd			
<u>D</u> escription:	P13		Browse	5			
EXT=IP/	AD			Dele <u>t</u> e			
LEVEL=XILINX LIBVEB=200				Mov <u>e</u>			
				Display All			
				<u>Clear Display</u>			
				Display File			
				Symbol Editor			
Apply	<u>0</u> K	Cano	cel	Help			
The assigned pin location will now appear at the bottom of the list of parameters for this IPAD symbol. The two dots in the left-hand margin indicate that the parameter name and value will appear in the schematic window attached to the IPAD symbol.

Symbol Prope	erties				×
Name:		-	Mov	e Name	
Beference:	In AP		Move F	Beference	1
Section:		_		ibutes	1
<u></u>	NO SECTIONS		<u></u> m	ibutes	
<u>F</u> ootprint:		*	<u> </u>	nments	
<u>T</u> echnology:		-	Pin Pa	rameters	
Parameters:					
<u>N</u> ame:	LOC	<u> </u>	Change		
<u>D</u> escription:	P13		Browse		
EXT=IP	AD			Dele <u>t</u> e	1
LEVEL:	⊧XILINX ⊧=2.0.0			Mov <u>e</u>	1
●● LOC=P	13			<u>D</u> isplay All	i
			C	lear Display	1
)isplay File	i
			S	ymbol Editor	j
Apply	<u>0</u> K	Cano	el	Help	

Double-clicking the parameter name-value pair removes one of the dots. This indicates that only the parameter value will appear in the schematic window. (Double-clicking again removes all the dots so neither the name or parameter value appears.)

Symbol Prope	erties				×
Name:	IPAD	•	Мо <u>v</u> а	Name	
<u>R</u> eference:	, CLK	_	<u>M</u> ove F	leference	
Section:	NO SECTIONS	-	Attri	butes	
<u>F</u> ootprint:		-	C <u>o</u> m	ments	
<u>T</u> echnology:		-	Pjn Pa	rameters	
Parameters: -	,				
<u>N</u> ame:	LOC	•	Change	Add	
Description:	P13		Browse		
EXT=IP	AD			Dele <u>t</u> e	
LEVEL=	XILINX -200			Mov <u>e</u>	
LOC=P1	3			<u>)</u> isplay All	
	~			ear Display	
				isplay File	
			Sy	mbol Editor	ĺ
Apply	ОК	Cano	cel	Help	

Click on OK to remove the **Symbol Properties** window. The named IPAD symbol appears in the **Schematic Editor** window along with its pin assignment.



We can repeat this process to set the name and pin assignment for the upper and lower OPAD symbols to (**S0**, pin 25) and (**S6**, pin 19), respectively. These assignments connect the **CNT7** and **CNT6** signals to the top and bottom segments of the LED digit on the XS40 Board, respectively. The pin reference names and pin assignments are awkwardly placed on the schematic, so we can reposition them by right-clicking on an OPAD symbol and selecting either Move Symbol Reference or Move Symbol Parameters from the pop-up list, respectively.



Once the pin names and assignments are positioned correctly, we need to change the net names of the wires that connect to the input and output pins so they match the names of the pins. This is not absolutely necessary, but it will help us later when we must interpret the report files generated by the implementation tools. To rename the clock net attached to the **CLK** input pin, just right-click on the wire and select Rename Net... from the pop-up menu.



The **Rename Net** window that appears shows the automatically-generated name of the clock input net (**\$NET00002_**). As shown below, we enter the replacement net name (**CLK**) and then click on OK.

Rename Net	×
Replace all instances of net for current	sheet:
\$NET00002_	ОК
with net:	Cancel
CLK	Help

Once the **Rename Net** window disappears, the new name for the clock input net appears in the **Schematic Editor** window. The process can be repeated to rename the nets attached to the **S0** and **S6** output pins as shown below.



Creating, Checking, and Exporting the Netlist

Now that we have the logic circuit constructed and all the nets and pins named and assigned appropriately, we can begin the process of creating and exporting a netlist that can be used by the implementation tools to compile the circuit for the XC4005XL FPGA. Start this phase by selecting the Options-Create Netlist menu item.



The netlist for the logic circuit will be generated in a few seconds and a window will appear informing us that the netlist was created but some possible trouble spots exist.

Schemat	ic Editor 🔀
٩	Netlist created successfully Warnings detected. Please inspect Project Manager window for details

We can examine the list of warnings in the **Command History** pane of the **Project Manager** window. Scrolling through the list shows that they all relate to outputs from the counters that we don't need to use for this particular design. So the warnings can be ignored in this case.

🐌 dsgn2_1 - 4005XLPC84-3 - Proje	ect Manager	_ 🗆 🗵
<u>File D</u> ocument <u>V</u> iew <u>P</u> roject <u>I</u> mplen	nentation <u>T</u> ools <u>H</u> elp	
D 🕞 🖯 🔍 🛤 🔍 🐂		
Files Versions Grade dsgn2_1 Grade dsgn2_1 Grad dsgn2_1 Grade dsgn2_1 Grade dsgn2_1 Grade	Flow Contents Reports Synthesis dsgn2_1 (4005XLPC84-3)	
Sc : Loadless net: CNT1 Sc : Loadless net: CNT0 Sc : Sc : Warning: unconnected inpu Sc : Unconnected output pin: Q[Sc : Unconnected output pin: TO Sc : Unconnected output pin: TO Sc : Unconnected output pin: CE Sc : Unconnected output pin: CE Sc : Unconnected output pin: TO	t and output pins detected 15:0] (\$12) > (\$12) EO (\$13) > (\$13)	-
Sc : Netlist created successfully Console	<u>र</u>	• •
Ready		

Once the netlist is generated, we can select the Options \rightarrow Integrity Test menu item to run a sequence of checks on the logic circuit.



The integrity test will complete in a few seconds and once again we will be informed that some potential problems were found. Checking the **Command History** pane shows the same set of errors found during the netlist creation step, so we don't have to be concerned.



Now that the netlist for our logic circuit has been generated and checked, we can export it in a format that can be used by the implementation tools. Select the Options \rightarrow Export Netlist... menu item to do this.



The **Export Netlist** window appears which allows you to select the directory and filename for the netlist. By default, the netlist file is placed in the top directory of the project with the same base name as the schematic file (dsgn2_1). There are three possible types for the exported netlist: Electronic Data Interchange Format (EDIF), XILINX Netlist Format (XNF), and as VHDL source code. EDIF is the preferable format in the majority of cases. Click on the Open button to create the EDIF netlist for our logic circuit.

Export Netlis	t				? ×
Look jn: 🔂	dsgn2_1	- 🗈	<u></u>	d	
🗋 lib					
🚞 xproj					
🔊 dsgn2_1.a	lb				
I					
File <u>n</u> ame:	dsgn2_1.alb				<u>O</u> pen
Files of <u>t</u> ype:	Edif 200 [*.EDN]		-		Cancel
	,		_		

Running Simulations with the Schematic Editor

The netlist has been exported, so we can now implement the design and test it in the FPGA on the XS40 Board. But we might want to simulate the circuit before doing that just to make sure it works. The problem with runningna simulation is the counter requires $2^{24} = 16,777,216$ clock cycles to go through all possible states. This would take a long time, so it might be more efficient just to test the upper byte of the counter to make sure that works as expected. In order to ease the interpretation of the simulation results, we need to rename the 8-bit counter. Start the renaming step by double-clicking the CC8CE counter symbol.



In the **Symbol Properties** window that appears, replace the automatically-generated name in the Reference field with CNTR. Then click on OK.

Symbol Prope	erties				×
Name:	CC8CE	•	Mo	1	
<u>R</u> eference:			Move	e Reference	
Section:	NO SECTIONS	-	A	<u>\</u> ttributes	
Eootprint:		-	C,	omments	
<u>T</u> echnology:		-	Pjn I	Parameters	
Parameters:	,				
<u>N</u> ame:		•	Chang	e <u>A</u> dd	
Description:			Brows	e	
LEVEL=	=MXILINX			Delețe	
LIBVER	=2.0.0			Mov <u>e</u>	7
				Display All	7
				<u>C</u> lear Displa	ע
				Display File	
				Symbol Edit	or
Apply		Cano	cel	Help	

The new name for the 8-bit counter now appears in the **Schematic Editor** window. Now click on the Simulation toolbox toolbar button to begin setting-up the simulation.



In the **SC Probes** window that appears, click on the Probe tool button that allows us to select various points to monitor in the circuit.



With the Probe tool active, click on the TC output of the 8-bit counter. A grey box will appear to indicate we have attached a probe to this output.





Add more probes to the CEO and Q[7:0] outputs of the counter.

In addition to the probes, we need to apply a stimulus to the inputs of the counter so it will do something. Click on the Stimulator tool button to initiate the adding of stimulators.



Now click on the clock and clock-enable inputs of the counter to add stimulators to these inputs. The logic signals applied by the stimulators will override any logic levels from the clock buffer and 16-bit counter outputs already attached to these inputs.



Once the counter's inputs and outputs are set-up, click on the Simulator button to bring up the **Logic Simulator** window.



The **Logic Simulator** window lists the various inputs and outputs to which we attached the stimulators and probes. Note that each input and output is prefixed with the reference name we assigned to the 8-bit counter. If we had not renamed the counter, it would have been more difficult to associate the various inputs and outputs with the probe points in the **Schematic Editor** window, especially in more complicated designs where we might monitor more than a single component.

🏶 Logic Simulator - Xilinx Foundation F2.1i [dsgn2_1] - [Waveform Viewer 0]
Eile Signal Waveform Device Options Tools View Window Help
🗃 🖬 🎒 🗞 🜮 🕢 Functional 🔄 👼 🔊 100ns 💽 🎯 Break 💽 🗛 🙀
2ns/div 20ns 40ns 60ns 80ns 100ns 120ns 140ns 0.0
OCNTR.TC
CNTR.CEO – .
BCNTR.Q7.(hex]
iCNTR.CE –
iCNTR.C –
0.0

We still need to describe the stimulus that enters the circuit through each stimulator. Select the Signal \rightarrow Add Stimulators... menu item. Then highlight the CNTR.C input and click on the B0 bit of the binary counter Bc in the **Stimulator Selection** window. This sends a clock waveform to the clock input of the CNTR counter.



We will place a manual stimulator on the clock-enable input of the counter. Highlight the CNTR.CE input and then click on the Q key in the Stimulator Selection window. Now the logic level on the CNTR.CE input will toggle whenever we press the **Q** key on the keyboard. Then click on Close to remove the **Stimulator Selection** window.



Press **Q** several times and watch the level on CNTR.CE toggle from logic 0 to 1 and back. Leave the clock-enable set to 0 to disable the counter. Then lcick on the Simulation Step toolbar button to run a few clock cycles through the counter.



Note that the counter value remains at 0x00 and does not increment as the clock pulses because the clock-enable input is held low.



Now press the **Q** key to toggle the clock-enable input to a logic 1. Then run another simulation step. Now we see the 8-bit counter increment its value on the rising edge of every clock cycle.



Every 100 ns simulation step applies ten clock cycles to the counter. We could click on the Simulation Step button 26 times in order to generate 260 clocks and observe the counter rollover from 0xFF to 0x00. But it's easier to change the duration of each simulation step by selecting 1 μ s from the pulldown list as shown below. Now we only need to run three simulation steps to rollover the counter.



After three clicks on the Simulation Step button, we can use the slider at the bottom of the **Waveform Viewer** window to bring the counter rollover event into view. Note that the clock-enable output of the counter (CNTR.CEO) becomes active on the cycle before the rollover. This output could enable another counter so that it increments once when the rollover of this counter occurs. This is the behavior we want to get by connecting the CEO of the 16-bit counter to the clock-enable of the 8-bit counter. By observing the operation of the CEO in the 8-bit counter, we gain some measure of assurance that the 16-bit counter will operate similarly. So the 8-bit counter should only increment when the 16-bit counter rolls over.

斄 Logic Simulator - Xilinx Foundation F2.1i [dsgn2_1] - [Waveform Viewer 0]	
🚾 <u>F</u> ile <u>S</u> ignal W <u>a</u> veform <u>D</u> evice <u>O</u> ptions <u>T</u> ools <u>V</u> iew <u>W</u> indow <u>H</u> elp	_ & ×
🖆 🖬 🎒 🖏 📶 Functional 🖃 🚮 💉 1us 💽 🎯 Break	• कं कं
	3us
2ns/div 12.6us 2.62us 2.64us 2.66us 2.68us 2.7us 2.7us 2.594us 1000000000000000000000000000000000000	tus 2.74us
OCNTR.TC	
OCNTR.CEO	
BCNTR.Q7.(hex 2 / FA / FB / FC / FD / FE / FF / 00 / 01 / 02 / 03 / 04 / 05 / 06 / 0	7 <u>(08 (09 (0A (</u>
iCNTR.CEq	
	╶╌╴Ӻ╌┠╌╌┠╶┠╌╌┠
Counter rollover event Move the slider	
until the rollover	
event appears.	
	•

We can view the logic levels on the various inputs and outputs by clicking the mouse in the **Waveform Viewer** window. This causes a cursor-line to appear and the value of each signal is shown at the left-hand side of the window. You can also view the signal levels attached to the probes in the **Schematic Editor** window.



Implementing, Downloading, and Testing the Counter Circuit

Once we have validated that the circuit should work, we can return to the **Project Manager** window. Since we successfully exported the netlist, there are green checkmarks by the schematic filename in the **Project Hierarchy** pane and in the Design Entry block of the **Project Flow** pane. Clicking on the Implementation block will initiate the compilation of the netlist into a bitstream for the FPGA.

File Descrete Monte Descrete Inclusion Table Hale
Tile Rocrumeur Alex Elolect Tubleweutation Tools Helb
Files Versions Image: Simplify and the second se
Conv : Export netlist 'C:\PRAG21I\DSGN2_1\dsgn2_1.alb' to EDIF 200 Conv : Total number of instances: 113 Conv : Total number of nets: 208 Conv : EDIF netlist exported to file - C:\PRAG21I\DSGN2_1\dsgn2_1.edn Pcm : EXIT: Schematic Editor

The **Implement Design** window will appear. There is no need to change any of the options from their default values for this design, so click on the Run button.

Implement Design
Device 4005×LPC84 Speed 3 -
⊻ersion name: ver1
<u>R</u> evision name: rev1
Control Files: <u>S</u> et
<u>Run</u> OK <u>C</u> ancel <u>H</u> elp

The tools should run uneventfully through all five phases of implementation process.

📽 dsgn2_1 (ver1->rev1)	- Flow Engine					
<u>F</u> low <u>V</u> iew <u>S</u> etup <u>U</u> tilitie:	s <u>H</u> elp					
<u> </u>	<u></u>					
XC4000XL Design Fl	ow (rev1)				Statu	s: 0K
	• 🎦 •		⇔	* ► ★ ★	⇔	
Translate	Мар	Place&Route		Timing (Sim)		Configure
Completed	Completed	Running				
Number of RPM Total equivalent Additional JTAG g Writing design fi Removed Logic Sum 3 block(s) rem 3 block(s) opt 1 signal(s) re	macros: 2 gate count for desi ate count for IOBs: le "map.ncd" mary: oved imized away moved	gn: 449 144				×
See MAP report fi	l. le "map.mrp" for de	tails.				
par -w -ol 2 -d	0 map.ncd dsgn2_1.n	cd dsgn2_1.pcf				
						<u>·</u>
For Help, press F1				XC400)5XL-3-PC84	dsgn2 1.ucf

The success of the implementation will be reported. Click on OK to return to the **Project Manager** window.

Project Manager 🛛 🗙
Flow Engine ver1->rev1 Completed Successfully.
()

Note the green checkmark in the Implementation block in the **Project Flow** pane, signaling a sucessful compilation of a bitstream for the FPGA. We can click on the implementation report browser button as shown below to peruse some of the statistics about this circuit.

シ dsgn2_1 - 4005XLPC84-3 - Proje	ct Manager	_ 🗆 🗵								
<u>File D</u> ocument <u>View Project</u> Implementation <u>T</u> ools <u>H</u> elp										
Files Versions G G dsgn2_1 G dsgn2_1 G dsgn2_1 G simprims C xc4000x	Flow Contents Reports Synthesis ver1 rev1 (XC4005XL-3-PC84) DESIGN ENTRY DESIGN ENTRY DESIGN ENTRY DESIGN ENTRY VERIFICATION VERIFICATION	j								
Xie : Flow Engine: ver1->rev1 (Ma	apped OK)	-								
Xie : Flow Engine: ver1->rev1 (Ri	outed OK)									
Xie : Flow Engine: ver1->rev1 (In	inieu OK) iniemented OK)									
Xie : Flow Engine ver1->rev1 Col	mpleted Successfully.	-								
Console	T	F								
Opens the Xilinx Report Browser										

Click on the Pad Report icon in the **Report Browser** window so we can check whether our circuit's inputs and outputs were assigned to the correct pins.

🔀 Report Br	owser - dsgni	2_1(ver1->rev1	1]	×
	\square			
Translation Report	Map Report	Place & Route Report	Pad Report	Asynchronous Delay Report
Post Layout Timing Report	Bitgen Report			

In the pad report file, we can see that the inputs and outputs were assigned as we directed.

Next we can check the timing statistics of the placed-and-routed circuit by clicking on the Post Layout Timing Report.

📆 Report Bro	owser - dsgni	2_1(ver1->revi	1)	×
Translation Report	Map Report	Place & Route Report	Pad Report	Asynchronous Delay Report
Post Layout Timing Report	Bitgen Report			

Near the bottom of the timing report file we find a summary for the counter circuit:

```
Timing summary:
-----
Timing errors: 0 Score: 0
Constraints cover 326 paths, 38 nets, and 54 connections (100.0% covera
Design statistics:
   Minimum period: 23.081ns (Maximum frequency: 43.326MHz)
   Maximum net delay: 8.110ns
```

Note that the maximum clock we can apply to this circuit is 43 MHz. Using a higher frequency could lead to incorrect operation depending upon factors like temperature and supply voltage variations.

We can drag-and-drop the bitstream file from the top-level directory of the project into the GXSLOAD window.

	Image: dsgn2_1 Image: state st
gxsload X Drop.BIT, .SVF, .HEX, and .EXO Exit files here to download to the XS or XSV Board. Recent Files: dsgn2_1.bit	ib is exp_EDIF.log xproj is logiblox.ini dsgn2_1.alb ime_sim.edn dsgn2_1.bit VHDL.LST dsgn2_1.EDN xproj.ini dsgn2_1.ll ime_sin_edn dsgn2_1.ll ime_sin_edn dsgn2_1.ll ime_sin_edn dsgn2_1.ll ime_sin_edn dsgn2_1.ll ime_sin_edn dsgn2_1.ucf ime_sin_edn dsgn2_11.SCH ime_sin_edn
Fileoad □ EEPROM Port LPT1 ▼	1 object(s) selected

The bitstream for the counter circuit will download into the FPGA on the XS40 Board and start to operate. If the oscillator on the XS40 Board is programmed for 50 MHz (the default frequency), then the upper segment of the LED digit will blink approximately three times per second while the lower segment blinks six times per second.

Why does the circuit work at 50 MHz when the timing report listed the maximum operating frequency as 43 MHz? Probably because your XS40 Board is sitting at room temperature with a solid, lightly-loaded supply voltage. If the temperature was increased to 70° C and the supply voltage drooped to 3.0V, then you might see the counter missing a beat. The only way to guarantee the circuit works across all variations of temperature and supply voltage is to decrease the input clock frequency and/or redesign the counter circuit.

Examining the Counter circuit with the FPGA Editor

The counter circuit was implemented in the FPGA and it works as planned, but exactly how are the pieces of the circuit *arranged in the FPGA*? We can find out by activating the FPGA Editor as shown below.

🐌 dsgn2_1 - 4005XLPC84-3 - Project Mar	nager	- 🗆 ×
<u>File D</u> ocument <u>View</u> Project <u>Implementatio</u>	on <u>T</u> ools <u>H</u> elp	
	Design Entry Simulation Marillantian	
Files Versions Flow □ □ dsgn2_1 □ □ dsgn2_1 □ □ dsgn2_1 □ □ simprims □ □ xc4000x	Implementation Constraints Editor Device Programming Elow Engine Utilities Flow Engine FPGA Editor PLD ChipViewer DESIGN ENTRY Lock Device Pins View Locked Pins Report Verification Implementation Implementation	rt –
Pcm : Reading Xilinx project Pcm : Opening project: pilitst		_
Pcm : Xilinx server initialization		۔
Runs the FPGA Editor		

The FPGA Editor operates on *Native Circuit Description* (NCD) files that list the locations of circuit elements and nets within the array of CLBs and routing resources of the FPGA. Your project may contain several NCD files, but you should usually select the one with the same name as the project as follows.

Multiple NCD Files	×
The directory that contains your design has more than one NCD file in it. One of the other NCD files may represent your probed design that you saved earlier. Select the NCD file you want to open.	
dsgn2_1.ncd map.ncd	

The **FPGA Editior** window will appear with three subwindows in it. The **Array** window shows the layout of the circuit components within the 14×14 array of CLBs in the XC4005XL FPGA. The **List** window lists all the components in the circuit. And the **World** window shows the area of the chip that is currently displayed in the **Array** window (this is most useful for orienting yourself when you have zoomed-in on the circuit in the **Array** window).

🎕 FPGA Edi	tor - da	sgn2	_1.ncd	I																						_ 8 ×
<u>File E</u> dit <u>V</u> i	ew <u>I</u> a	ols	<u>W</u> indov	v <u>H</u> elp)																					
	8	8	E	1 🖻					X	₽l		Ø		?	- -	– }- -•	- 🔠 🍨	₽ [A	<u>,</u>				
St Arrav1		_			_				_									90 L i	st1	_	_				×I	cxit
																									=1	add
																		All C	Compon	ents				-		attrib
	<u>ک</u> ر																		N	ame	Туре	#Pins	Select	Hilited		autoroute
							H											1	\$12	2/Q<1	CLB	7	No			clear
		ы	в.	ы.		В.	ы	B	ы	ы	В.	ы.	В.	в.	в			2	\$12	2/Q<1	CLB	7	No		_	delay
		Ы		E]:	Ц	ы	Б	₫0	đ	E.	Е		Н	Ц	ы			3	\$12	2/Q<8	CLB	7	No			delete
				-														4	\$0	2/Q<1	CLB	7	No		_	drc
			T	ы.	ч	T T	3	ч П	E.	ш,	T	ч.	T	T	T			6	\$0	2/Q<0	CLB	6	No		_	editblock
		LŢ		d,	п.	B	d	di.	d			ď	B		d			7	\$L	2/Q<6	CLB	7	No			find
		Π.																8	\$12	2/Q<4	CLB	7	No		_	hilite
	- 16 - I	<mark>اب</mark>		미.	뎍			¤ :				9						9	\$12	2/\$113	CLB	2	No			info
	16	L I		EC						EI.								10		11<4> IT<6>	CLB	8	NO		_	probes
		1																12		IT<0>	CLB	7	No			route
		1	đ	ď.	đ	Ľ	đ					ď.						13	Ch	IT<2>	CLB	8	No		_	swap
		1			. L													14	CL	K.	CLKIOB	1	No			unroute
		F.	Ц	다.	đ	Ц	5	Ħ.	đ,	۵,	Ľ	а;	Ц	П	П			15	SC	1	ЮВ	1	No			
			Ц	- =	Ц	E	8		El	E.		d.	Ш	Ц	E			16	SE	-	IOB DUFOI	1	No			
		h						HI										17	21:)	BUFGL	2	INO			
	E	۳2.	9	<u>я;</u>	đ	8	9	F I	Ø,	E.		9	9	9	Ē											
		Ľ		El:			e	el:	EL.	Ē	B	e:	B					<u> </u>								
							<u> </u>											N W	orld1						×1	
		E.	E	ď.	đ	đ	đ		Ľ.	Ľ	Ľ	d.	Ľ	Ľ	Ľ				-				_			
		Ξ.	5	Щ.	5	Ц	5	S	E,	Ξ.	5	Ч.	5	5	E.											
				а:				۹::				а:														
Building	chin	ar	anhic	~																					_	
Duriding	chip	gr.	aphic																							=
																										<u> </u>
 						_	_		_			_	_											05.1- 1	14.0	Deed Out
For Help, pre	ess F1																						40	102xibc8	54-3	Read Unly

You can click on entries in the **List** window and the associated element in the **Array** window will be highlighted. Below, I have clicked on the **CLK** input buffer in the **List** window and red squares have appeared in the upper left-hand corner of both the **Array** and **World** windows to indicate the location of this IOB in the FPGA. (The red square in the Array window is very small, so you may have to zoom-in a bit to see it.)

Step FPGA Example 2	ditor -	dsgn2	_1.ncc	l																			
<u>File E</u> dit <u>\</u>	/iew _	<u>T</u> ools	<u>W</u> indov	∾ <u>H</u> el	P																		
		*	₿E E	1 🖻					E	۶C] I	P		?		┛	- ::: 🗘	의문	🔀 🖸 A 🔮	L			
😵 Array1																	- 	😵 List				- 🗆 🗵	exit
																							add
					ı k		D											All Co	mponents			<u> </u>	attrib
	Ц						D												Name	Туре	#Pins	Select	autoroute
		B		Ø:	ਈ			BC	ਈ	El	B	e:	Ø		El			1	\$I2/Q<12>	CLB	7	No	clear
																		2	\$12/Q<14>	CLB	7	No	delay
		E	Ľ	[]		Ш		B	Ľ	E.	Ľ		Ы	Ц				3	\$12/Q<8>	CLB	7	No	delete
				n°				n ^{(**}										4	\$12/Q<10>	CLB	7	No	drc
			I	ч.	I	T	T		Ξ.	ш.,	Ĩ	ч.	I	I	I			6	\$12/Q<0>	CLB	6	No	editblock
			1.		۱. ۲			_										7	\$12/Q<6>	CLB	7	No	find
		T.	4	ч.	4	ы	ц	4	ы.	ы.	4	щ.		ч	4			8	\$I2/Q<4>	CLB	7	No	hilite
		⊢		Ф.	đ			9				Ф.						9	\$12/\$11307_CINt	CLB	2	No	info
																		10	CNT<4>	CLB	8	No	probes
		-	L.	Π.	۵		R	₿	а	R	Ц	Ш.	R.	R.	R.			11	CNT<6>	CLB	7	No	route
	. –	╧╧╉	D	с і .	a	Ci	Ľ.		, L	Ľ.	13	с. С	L.	L.	L.			12	CNT<0>	CLB	7	No	swap
					t	_	-4											13		CLKIOR	8 1	NO	unroute
		_		ц.			E -	-				ه						15	S SO	IOB	1	No	
																		16	S6	IOB	1	No	
		₽ ₽	G	a :	Ш	E	8		E .	E.	Ш	9	Ш	Ш	Ш			17	\$15	BUFGL	2	No	
			d	d.			, B		d	d	d	01											
		TH-																				F	
		÷	면	Ø.	E	E	Ø	el:	던	린		e:						J —					
																		🕸 Wor	ld1			<u>- 🗆 ×</u>	
		E	Ľ	¤ 1:	Ц	Ш	В	₫	데	EI.	Ľ	1	Ľ	Ц	Ľ								
				đ	El							d											
							-				-												
				뎍.				⊑:				며:											
Decid all	1		1.		_	_		_		_	_		_	_									
Building	chi	p gr	aphic	s																			
For Help	iess F	1																			40	05xlnc84	-3 Bead Oplu
. or meip, p		•																			140	oompoor	o jnosa only

You can highlight a set of components in the **List** window by clicking on the first element of the set and then shift-click on the last element. Doing this with the 16-bit counter (whose internal reference name in the schematic is **\$12**) shows that the counter is arranged as a column of 8 CLBs with each CLB containing 2 counter bits.



You can also select non-adjacent components by ctrl-clicking on different list entries. The **CLK** input pad, clock buffer, and least-significant bit of the 16-bit counter are selected below.

🐏 FPGA Editor - dsgn2_1.ncd										
<u>File Edit ⊻iew Tools W</u> indow <u>H</u> elp										
	፼፼ <u>₿₽□</u> ₿₽₿	🛯 🕂 <u></u>	\$P\$\$\$ <u>∎ A</u> ₹							
			🛿 List1 📃 🔲	× cxit						
Clock bullet				add						
		×	All Components	attrib						
			Name Type #Pins Select	autoroute						
			1 \$12/Q<12> CLB 7 No	clear						
			2 \$12/Q<14> CLB 7 No	delay						
			4 \$12/Q<10> CLB 7 No	dto						
	वःः व व व		5 \$12/Q<2> CLB 7 No	editblock						
CLK IOB			6 \$12/Q<0> CLB 6 Yes	find						
::: 📥 ધ ચ. હા હ હ	वः व व व वः	a a a 201	7 \$12/Q<6> CLB 7 No 8 \$12/Q<6> CLB 7 No	hilite						
	व्यांव व्यंव्यं		9 \$12/\$11307_CINt CLB 2 No	info						
			10 CNT<4> CLB 8 No	probes						
			11 CNT<6> CLB 7 No	route						
	💻 व व व व	d d d b) (12 CN1<0> CLB 7 NO	swap						
	(14 CLK CLKIOB 1 Yes	unroute						
ंः 📻 व च च व व व		a a a sei	15 S0 IOB 1 No							
	= = = =		16 S6 IOB 1 No							
			17 \$15 BUFGL 2 Yes							
ः 📥 व स स स स	र्घ व व र र	a a a 🚌 : 📕	<u></u>	·						
			🔆 World1 📃 🗖	×						
	et.) et et et.	d d d às i								
LSB OF 16-										
bit counter										
	बःः ब ब ब बः									
comp "\$I2/\$1I307_CINt" in macro "\$I	2/hset", site "CLB_R3	C1", type = CLB.		<u>ب</u>						
For Help, press F1 4005xlpc84-3 Read (



You can look at the details of what is implemented in each CLB by double-clicking the CLB in the **Array** window.
The detailed routing within the CLB for the two least-significant bits of the 16-bit counter is shown below. Each CLB contains many possible routes between components (shown as purple lines), but only a few are activated for each particular function (shown as blue lines). The clock for the counter bits enters at ① and connects to the clock inputs for the counter flip-flops at ② and ③. The output from the least-significant counter bit comes out of the CLB at ④ but also feeds back into the CLB again at ⑤. The counter bit enters the lookup table at ⑥. The LUT just inverts the counter bit before sending it back into the data input of the flip-flop at ⑦. This makes the least-significant bit toggle back and forth from $0 \rightarrow 1 \rightarrow 0$... on each successive rising clock edge.



The second bit of the counter exits the CLB at ① and re-enters it at ②. The LUT accepts the counter bit along with the output of the carry-generation block at ③. The LUT will toggle the value of the second counter bit if the carry output at ③ is high. Then the updated counter bit enters the second counter flip-flop at ④. The result is that the second counter bit follows the pattern $0 \rightarrow 0 \rightarrow 1 \rightarrow 1 \rightarrow 0 \rightarrow ...$ on successive clock rising clock edges.

The carry-generation block is set to act as an incrementer since that is what a counter does. The carry input to the carry generator at (5) is forced to a logic 1 for the first counter stage so that the counter will increment. The carry generator also takes the values of the two counter bits at (6) and (7) to compute the carries for the second counter bit at (3) and the carry for the third counter bit at (8). The carry at (8) exits the CLB at (9) so it can be routed to the third counter bit in the next CLB.



For the rest of the counter bits, the carry output from the preceding CLB feeds into the carry generator of the next CLB at O. The carry input also feeds directly into the LUT of the lower counter bit within each CLB at O.



The carry output from the final bit of the 16-bit counter becomes the clock-enable for the 8-bit counter that follows. The carry output will only go to logic 1 when the 16-bit counter value is 0xFFFF, and this is what's needed to enable an increment of the 8-bit counter. The carry output from the final bit in the 16-bit counter enters another CLB at ① and passes through an LUT at ② and is output unchanged at ③. This output becomes the clock-enable for the 8-bit counter. This intermediate CLB is needed to make the interchange because there is no direct route from the carry output of the 16-bit counter to the clock-enable input of the 8-bit counter.



© 2001 by XESS Corp.

The implementation of the counter bits for the 8-bit counter is very similar to that of the 16-bit counter. The only difference is the clock-enable that enters at ① and passes through the multiplexer at ②. Then the clock-enable enters the flip-flops at ③ and ④ so the eight-bit counter only increments on the rising edge of the clock when the 16-bit counter has a value of 0xFFFF.



The outputs from the upper two bits of the 8-bit counter are output from the FPGA through IOBs. The output from the most-significant bit of the 8-bit counter enters the IOB at ① and passes through a multiplexer at ②. The output of the multiplexer goes through a buffer at ③ that drives a pin of the FPGA package at ④.



Building the Counter Circuit for the XS95 Board

The counter circuit can be built for an XS95 Board with an XC95108 CPLD by starting a new schematic-based project as shown in the following screen. (You can't just create a new version as we did in the last chapter because a completely different library of schematic symbols for the XC9500 CPLD needs to be linked to the project.)

New Proje	ect		×
Name:	dsgn2_2		ОК
Directory:	C:\PRAG21I		Cancel
Туре:	F2.1i	•	
Flow:		C HDL	
XC9500		4 🗸	20 💌

Then re-create the schematic using the library of XC9500 schematic symbols. Note that the pin assignments are different since this design is intended for the XS95-108 Board (see Figure 6).





Figure 6: Connection of the programmable oscillator and LED digit to the pins of the CPLD on the XS95 Board.

After creating and exporting the netlist for the counter circuit, activate the implementation tools to fit it into the XC95108 CPLD.

Implement Desi	gn				×
Device 95108	3PC84	•	Speed	20	•
Version name:	ver1				
<u>R</u> evision name:	rev1				
Control Files:	<u>S</u> et			<u>O</u> ption	IS
<u>[</u> un]	ОК	<u>C</u> ar	ncel	<u>H</u> elp	

The implementation tools should proceed through all four stages without incident.

🏪 dsgn2_2 (ver1->rev1) - Flow I	Ingine			
<u>Flow View Setup Utilities Help</u>				
💯 🗹 🖻 🐠 🕅				
XC9500 Design Flow (rev1)			Status: OK	
∎ ∎ ⊒				
Translate	Fit	Timing	Bitstream	
Completed	Completed	Completed	Completed	
writing EDIF file	to 'time_sim.edn'		<u> </u>	
xcpy time_sim.edn c:∖p	rag21i\dsgn2_2\time_sim.	edn		
hprep6 -i dsgn2_2 -r j	ed -a -1 dsgn2_2.log -n	dsgn2_2		
Programming File Gener	ator: version C.22	,		
(c) Copyright 1989-1997 Xilinx Inc. All rights reserved.				
xcpy dsgn2_2.jed c:∖pr	ag21i\dsgn2_2\dsgn2_2.je	ed	-	
			>	
For Help, press F1		XC	95108-20-PC84 dsgn2_2.ucf 🛛 🦯	

Once the schematic entry and implementation steps are complete, click on the Programming box in the **Project Flow** pane. Then create an SVF bitstream file for the counter circuit.



Finally, download the SVF file into the XC95108 CPLD on the XS95 Board. Once again, if the oscillator on the XS95 Board is programmed for 50 MHz (the default frequency), then the upper segment of the LED digit will blink approximately three times per second while the lower segment blinks six times per second.

	🔁 rev1	- 🗆 ×		
	<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>G</u> o F <u>a</u> vorites <u>H</u> elp	-		
		0-0- 0-0- 0-0-		
	Address C:\Prag21i\dsgn2_2\xproj\ver1\rev1			
State gxsload Drop .BIT, .SVF, .HE files here to downloa	 command.his dsgn2_2.rpt dsgn2_2.vf time_sim.edn dsgn2_2.ctl dsgn2_2.ucf xc9500.imp dsgn2_2.gyd dsgn2_2.vm6 dsgn2_2.ide dsgn2_2.ide dsgn2_2.xbt dsgn2_2.log fe.log dsgn2_2.mod itagpgmr.log dsgn2_2.ngd revision.obf 			
XS or XSV Board.	1 object(s) selected 🛄 My Com	puter /		
dsgn2_2.svf				
<u>R</u> eload	EEPROM Port LPT1 -			

We can examine the arrangement of the circuit components over the macrocells of the XC95108 CPLD by starting the CPLD ChipViewer tool as shown below.



The **ChipViewer** window appears with a right-hand subwindow that shows which macrocells and pins are used. The left-hand subwindow displays the collapsed hierarchy of the counter circuit. Click on the plus sign to expand the hierarchy.

File Edit View Help	
Solution State Sta	믜페
	ᆁ
Complete	

Expanding the **dsgn2_2** entity exposes the six functional blocks (**FB1–FB6**) of the XC95108 CPLD. Four of the functional blocks are empty, but **FB1** and **FB3** contain components of the counter circuit and can be expanded further. As shown below, the clock signal enters through **FB1** and the two outputs are generated by macrocells in **FB3**.

To examine the functions of the macrocells, click in the right-hand window and then click several times on the zoom-in button.



Poising the mouse cursor over on of the I/O pins or over a macrocell displays the particular options that are active for that object. For example, macrocell **MC3** in **FB3** is configured in a mid-level power-consumption mode and the flip-flop in the macrocell will be preloaded with logic 0 upon when the CPLD powers up. We can observe more details by double-clicking the macrocell.



A window appears that shows the details of macrocell MC3 in FB3 including the list of inputs that affect the macrocell and the associated logic equation.



For the most part, that's all you can do with ChipViewer. It doesn't yet have all the features you will find in FPGA Editor. If you like graphical displays and a point-and-click interface, then ChipViewer may be the tool for you. I prefer to just look in the fitter report file for the CPLD where all the same information (and more) is presented in a condensed format.