Creating a Core Component

Summary

Tutorial

TU0123 (v2.0) March 04, 2008

This tutorial is designed to give you an overview of how to create a core component, synthesize the EDIF and generate a schematic symbol of the core. It covers creating the core project, synthesizing, publishing and generating the symbol in the Schematic Library Editor.

Creating a core component offers the advantages of design reuse and design security. The design of the core component can be captured in schematic and VHDL and then synthesized to create the EDIF model. **EDIF** stands for **Electronic Design Interchange Format** and EDIF models store electronic *netlists*.

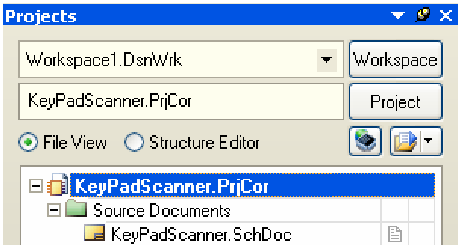
From the core project, you can also generate a new schematic symbol of the core component. By adding different configurations to the core project, the core component can then be used with different FPGA devices.

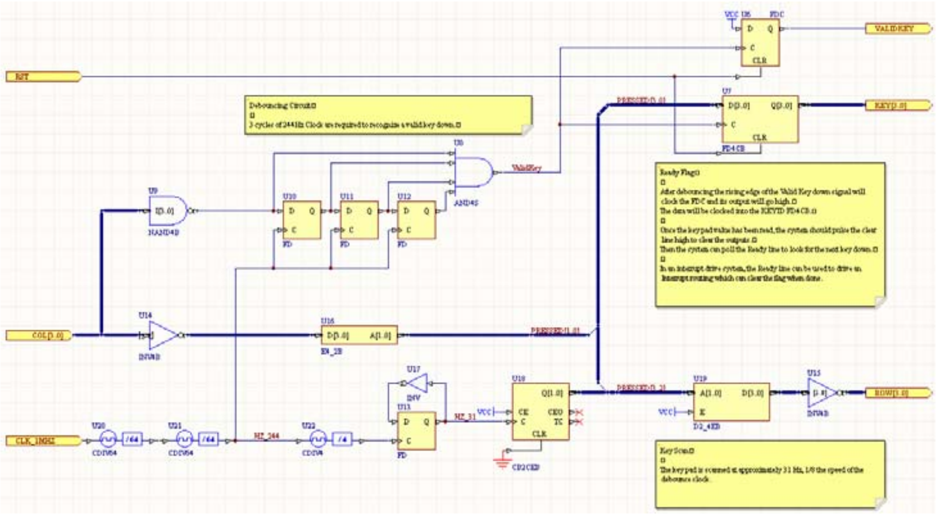
Creating a Core Project and Schematic

This tutorial is based around the creation of a core component for decoding the keypad on the NanoBoard-NB1.

1. Create a new core project by selecting **File » New » Project » Core Project**. Save the core project as `KeyPadScanner.PrjCor`, making sure the project name has no spaces in it.
2. Create or add the schematic that you wish to use as the core to the core project, e.g. `KeyPadScanner.SchDoc`, shown below in Figure 1. This schematic is available from the `\Examples\NB1 Examples\Processor Examples\I2C DAC and ADC - TSK165B` folder of the installation. Note that the top sheet of the core design must be the same as the core project name.

   To add this schematic to your core project, right-click on the project name in the Projects panel and select **Add Existing to Project**. Choose the schematic file, `KeyPadScanner.SchDoc`, and click **Open**. The schematic document is added to the project and appears in the *Projects* panel under **Source Documents**. Save the project.





Debouncing Circuit

3 cycles of 244Hz Clock are required to recognize a valid key down.

Ready Flag

After debouncing the rising edge of the Valid Key down signal will clock the FDC and its output will go high.

The data will be clocked into the KEYID FD4CB.

Once the keypad value has been read, the system should pulse the clear line high to clear the outputs.

Then the system can poll the Ready line to look for the next key down.

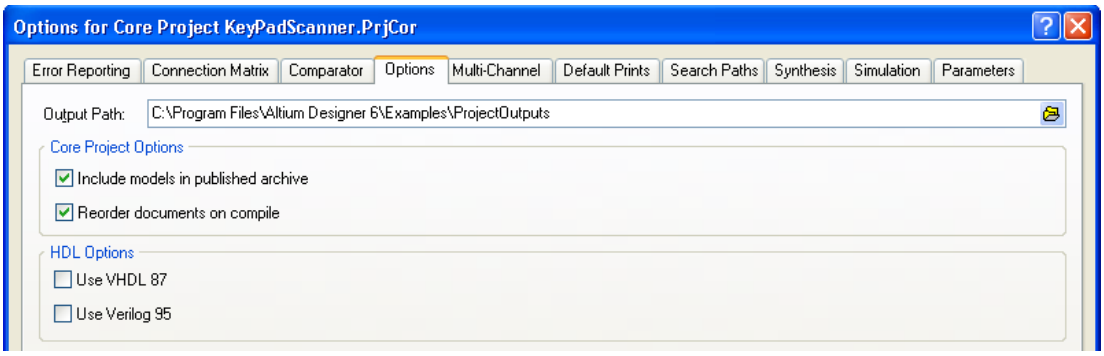
In an interrupt drive system, the Ready line can be used to drive an Interrupt routine which can clear the flag when done.

Key Scan

The keypad is scanned at approximately 31 Hz, 1/8 the speed of the debounce clock.

Figure 1. KeyPadScanner.SchDoc

3. Now we need to set up the project options. Select **Project » Project Options** and click on the **Options** tab of the *Options for Core Project* dialog.



Output Path: C:\Program Files\Altium Designer 6\Examples\ProjectOutputs

Core Project Options

☑ Include models in published archive

☑ Reorder documents on compile

HDL Options

□ Use VHDL 87

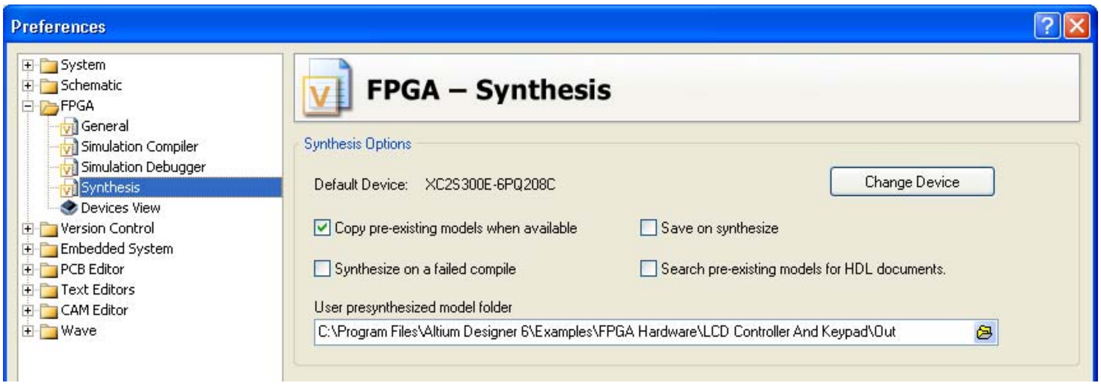
□ Use Verilog 95

Make sure the **Include models in published archive** option is selected. This will bundle all EDIF files generated for the core component together in a zip file when you 'publish' the core later. Click **OK**.

4. For these EDIF files to be generated, a User EDIF models folder must be created. In Microsoft Windows, create a folder, for example `MyCores`, which will hold your EDIF models. Whenever you synthesize any design, this folder is searched for models, along with the standard system EDIF folders (`\Library\EDIF`).

5. Now we need to specify this new folder name in the *FPGA - Synthesis* page of the *Preferences* dialog (**Tools » FPGA Preferences**) so the system knows where to look for your published EDIF files during synthesis.

   Click on the folder icon next to the **Use presynthesized model folder** option to browse to the folder's location, as shown in the dialog below. Click **OK**.



Synthesis Options

Default Device: XC2S300E-6PQ208C

☑ Copy pre-existing models when available

□ Save on synthesize

□ Synthesize on a failed compile

□ Search pre-existing models for HDL documents.

User presynthesized model folder

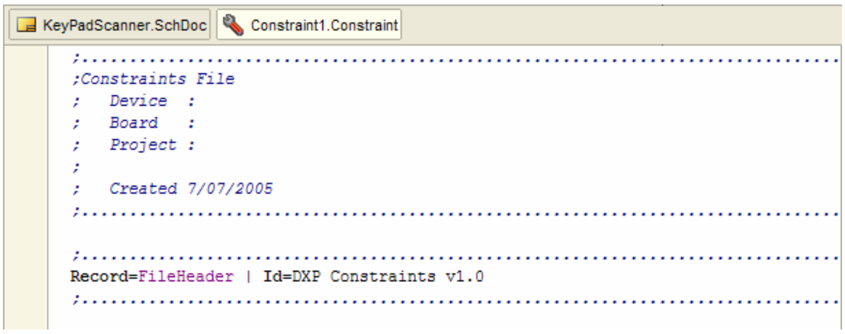
C:\Program Files\Altium Designer 6\Examples\FPGA Hardware\LCD Controller And Keypad\Out

Creating Configurations and Constraints

The next setup required is the creation of configuration and constraint files targeting the devices you want your core component to be used on. A constraint file will set the target FPGA device. Note that you need to create one configuration for each device family that you want the core to target. The core component can then be used in any device in that device family. We will set up two configurations so the core component can be used with two different target FPGA device families.

To create a new constraint file:

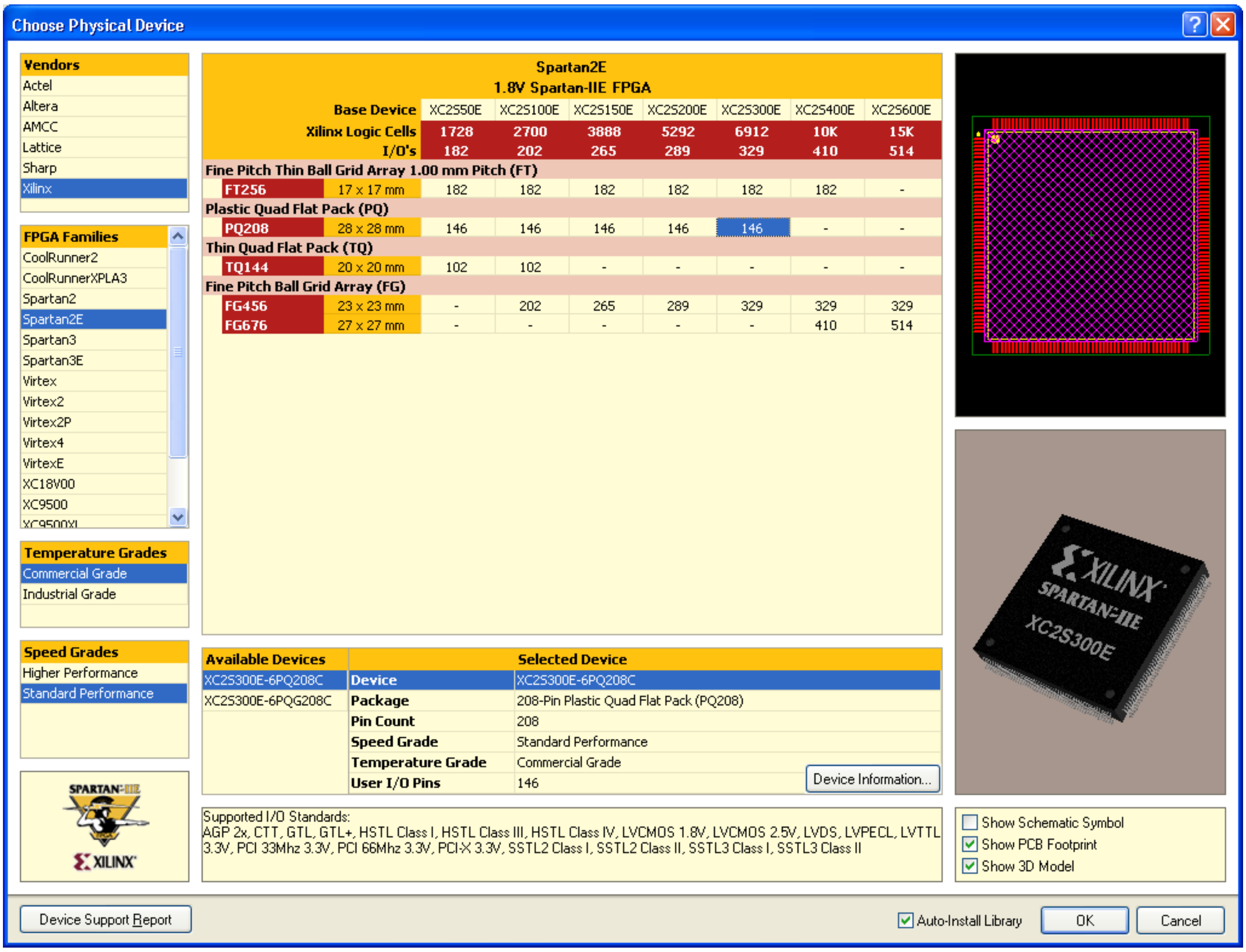
1. Right-click on the core project name in the **Projects** panel and select **Add New to Project » Constraint File**. A Constraint file will appear in the `Settings\Constraint Files` folder in the Projects panel and a new file, `Constraint1.Constraint`, opens in the Constraint editor.



```
;..............................................................................
;Constraints File
;   Device  :
;   Board   :
;   Project :
;
;   Created 7/07/2005
;..............................................................................

;..............................................................................
Record=FileHeader | Id=DXP Constraints v1.0
;..............................................................................
```

2. From the Constraint editor menus, select **Design » Add/Modify Constraint » Part**. The *Choose Physical Device* dialog appears.



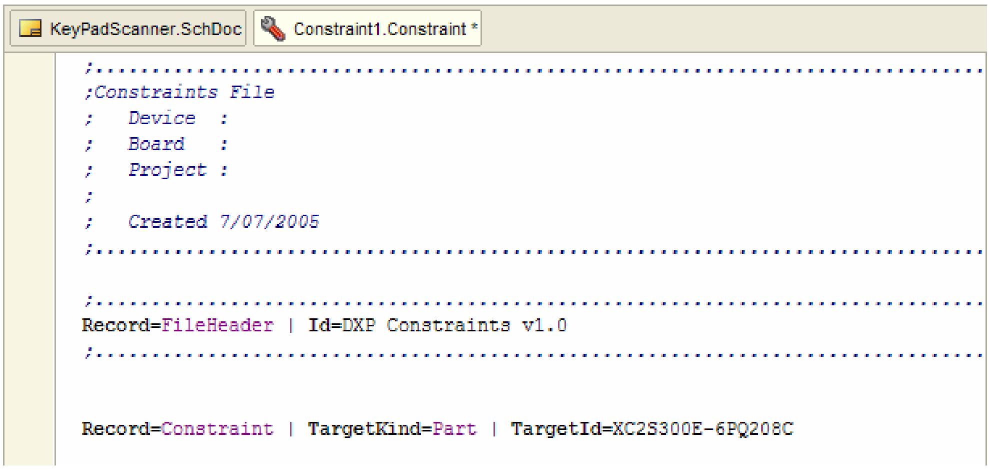
Spartan2E
1.8V Spartan-IIE FPGA

		Base Device	XC2S50E	XC2S100E	XC2S150E	XC2S200E	XC2S300E	XC2S400E	XC2S600E
		Xilinx Logic Cells	1728	2700	3888	5292	6912	10K	15K
		I/O's	182	202	265	289	329	410	514
Fine Pitch Thin Ball Grid Array 1.00 mm Pitch (FT)									
FT256	17 x 17 mm		182	182	182	182	182	182	-
Plastic Quad Flat Pack (PQ)									
PQ208	28 x 28 mm		146	146	146	146	146	-	-
Thin Quad Flat Pack (TQ)									
TQ144	20 x 20 mm		102	102	-	-	-	-	-
Fine Pitch Ball Grid Array (FG)									
FG456	23 x 23 mm		-	202	265	289	329	329	329
FG676	27 x 27 mm		-	-	-	-	-	410	514

Available Devices	Selected Device	
XC2S300E-6PQ208C	Device	XC2S300E-6PQ208C
XC2S300E-6PQG208C	Package	208-Pin Plastic Quad Flat Pack (PQ208)
	Pin Count	208
	Speed Grade	Standard Performance
	Temperature Grade	Commercial Grade
	User I/O Pins	146

Supported I/O Standards:
AGP 2x, CTT, GTL, GTL+, HSTL Class I, HSTL Class III, HSTL Class IV, LVCMOS 1.8V, LVCMOS 2.5V, LVDS, LVPECL, LVTTL 3.3V, PCI 33Mhz 3.3V, PCI 66Mhz 3.3V, PCI-X 3.3V, SSTL2 Class I, SSTL2 Class II, SSTL3 Class I, SSTL3 Class II

3. We will set up a constraint file to target the Xilinx SpartanIIE chip first. Click on **Xilinx** in the Vendors list and select **Spartan2E**. The SpartanIIE included with the **NanoBoard-NB1** is a 208 pin Quad Flat Pack package, with 300,000 equivalent gates. Click on the cell that represents this device, i.e. the **146** at the junction of XC2S300E and PQ208 for a XC2S300E-6PQ208. Click **OK** and a new device specification constraint is added into the Constraint file.



```
;..............................................................................
;Constraints File
;   Device  :
;   Board   :
;   Project :
;
;   Created 7/07/2005
;..............................................................................

;..............................................................................
Record=FileHeader | Id=DXP Constraints v1.0
;..............................................................................


Record=Constraint | TargetKind=Part | TargetId=XC2S300E-6PQ208C
```

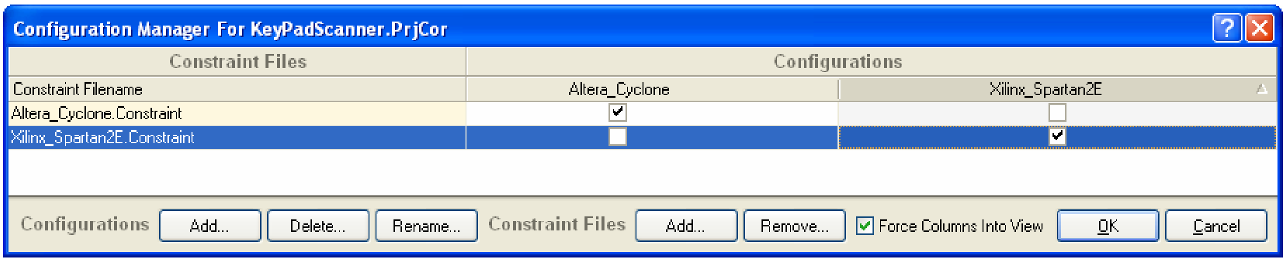
4. Save the constraint file as `Xilinx_Spartan2E.Constraint`.

5. Create another constraint file, called `Altera_Cyclone.Constraint` using steps 1-2 for the Altera Cyclone, 240 pin quad flat pack, 450,000 gate device (EP1C12). Save and close the constraint files.

6. To assign constraint files to a project, you must create a configuration for each unique output that you intend to generate. Add a configuration and constraints file to the core project by selecting **Project » Configuration Manager**. Add a

Configuration named `Xilinx_Spartan2E` by clicking on the **Add** button (next to Configurations) and click **OK**. Select the configuration check box.

7. Add another configuration called `Altera_Cyclone` and select the configuration checkbox to match the constraint file.



Constraint Files	Configurations	
Constraint Filename	Altera_Cyclone	Xilinx_Spartan2E
Altera_Cyclone.Constraint	☑	□
Xilinx_Spartan2E.Constraint	□	☑

☑ Force Columns Into View

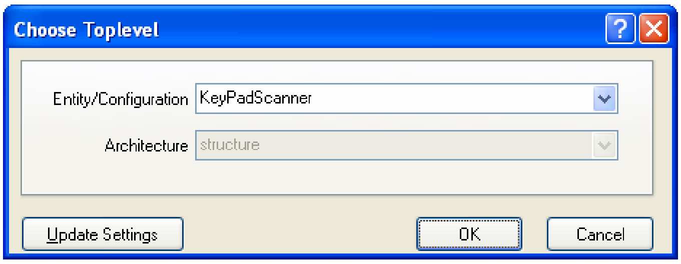
8. Click **OK**. Close the constraint files. Save the core project.

Synthesizing the Configurations

Next, we need to synthesize all the configurations that we have just set up.

1. From the Schematic Editor, select **Design » Synthesize All Configurations** to synthesize the core project, i.e. generate the intermediate VHDL files from the schematic(s) and then synthesize them into EDIF, ready for the FPGA vendor's place and route tools.

   If you have not already nominated a top level entity/configuration in the **Synthesis** tab of the *Options for Core Project* dialog (**Project » Project Options**), the *Choose Top-level* dialog will appear. Enter the core project name or select from the dropdown list. Click **OK** to continue.



Entity/Configuration: KeyPadScanner
Architecture: structure

   All configurations will be synthesized and the resulting intermediary VHDL files for the schematic, EDIF files for the schematic wiring and individual parts and a synthesis logfile will be generated and display under the `Generated (config_name)` folders in the *Projects* panel.

   If there are any error messages displayed in the *Messages* panel, go back to the schematic to fix any issues and resynthesize.

Publishing the Core

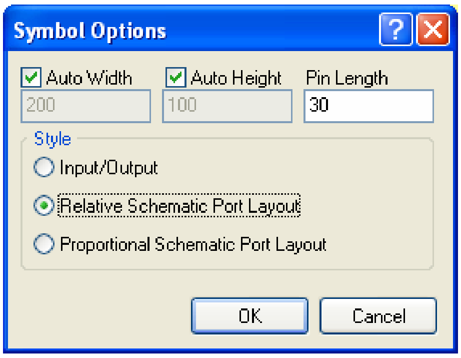
Now we can 'publish' the core project. This will zip together (archive) all the EDIF files in the core project's Project Outputs folder, and then copy this to the user EDIF models folder that you specified earlier.

1. Select **Design » Publish**. If the error message 'cannot find " " working folder' appears, make sure you have set up the **Use presynthesized model folder** option in the *FPGA – Synthesis* page of the *Preferences* dialog.
2. Check the *Messages* panel (**View » Workspace Panels » System » Messages**) if you wish to see that the publication has successfully taken place.
3. Save the core project file.

Creating the Core Component from the Schematic

Now that the core itself is ready (in the form of a set of EDIF models), the next step is to create a schematic symbol that represents the core component, and link it to the EDIF files generated during synthesis and archived together by publishing.

1. Select **Design » Generate Symbol**. For this example, we will create a new library, so confirm its creation by clicking **Yes** in the *Confirm* dialog. The *Symbol Options* dialog appears. Select **Relative Schematic Port Layout** to arrange the pins on the symbol in the same arrangement as the ports on the top-level schematic in the core project. Leave the other defaults selected and click **OK**.

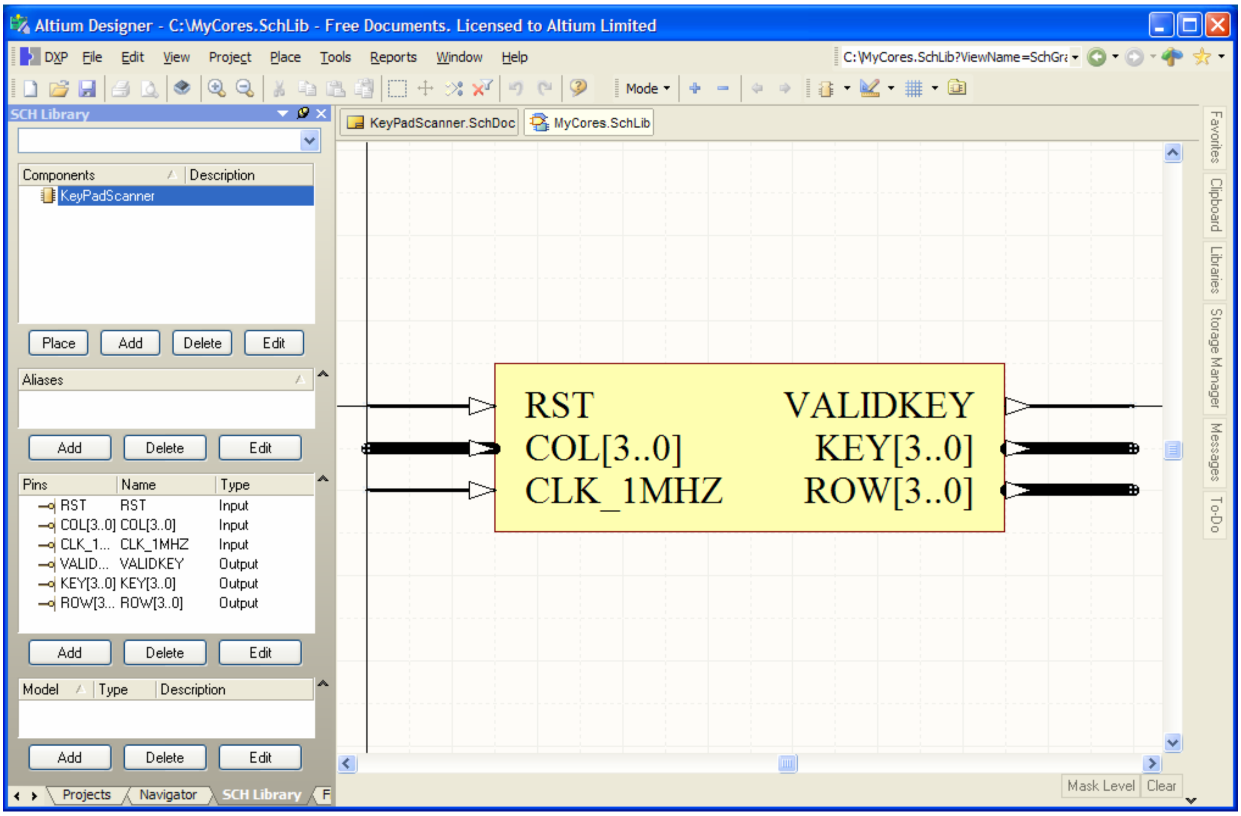


Symbol Options

☑ Auto Width 200
☑ Auto Height 100
Pin Length 30

Style
○ Input/Output
● Relative Schematic Port Layout
○ Proportional Schematic Port Layout

2. A new schematic library file (`Schlib1.SchLib`) opens and displays the generated symbol from the schematic called `KeyPadScanner`, using the ports from the schematic to create and name the pins, as shown in Figure 2 below. Save the new library as `MyCores.SchLib`.



Altium Designer - C:\MyCores.SchLib - Free Documents. Licensed to Altium Limited

C:\MyCores.SchLib?ViewName=SchGr

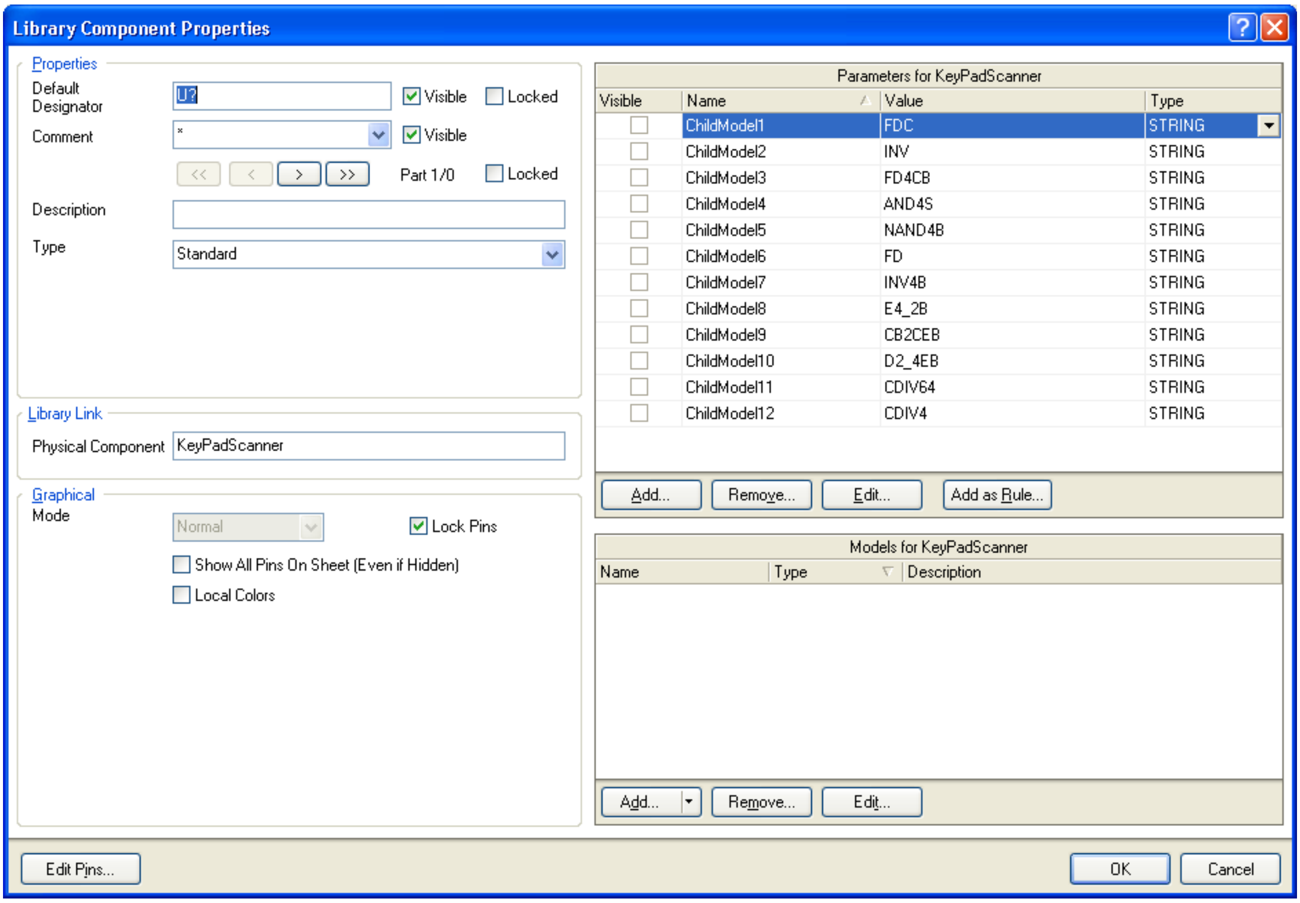
Components	Description
KeyPadScanner	

Pins	Name	Type
RST	RST	Input
COL[3..0]	COL[3..0]	Input
CLK_1...	CLK_1MHZ	Input
VALID...	VALIDKEY	Output
KEY[3..0]	KEY[3..0]	Output
ROW[3...	ROW[3..0]	Output

Figure 2. KeyPadScanner component in the Schematic Library Editor

3. Click on the SCH Library panel (**View » Workspace Panels » SCH » SCH Library**) to display the component's details.

4. Double-click on the component name in the SCH Library panel (or click on **Edit**) to display the *Library Component Properties* dialog.



Default Designator: U? ☑ Visible ☐ Locked

Comment: * ☑ Visible

Part 1/0 ☐ Locked

Description:

Type: Standard

Physical Component: KeyPadScanner

Mode: Normal ☑ Lock Pins

☐ Show All Pins On Sheet (Even if Hidden)

☐ Local Colors

Visible	Name	Value	Type
☐	ChildModel1	FDC	STRING
☐	ChildModel2	INV	STRING
☐	ChildModel3	FD4CB	STRING
☐	ChildModel4	AND4S	STRING
☐	ChildModel5	NAND4B	STRING
☐	ChildModel6	FD	STRING
☐	ChildModel7	INV4B	STRING
☐	ChildModel8	E4_2B	STRING
☐	ChildModel9	CB2CEB	STRING
☐	ChildModel10	D2_4EB	STRING
☐	ChildModel11	CDIV64	STRING
☐	ChildModel12	CDIV4	STRING

Name	Type	Description

Note that the parameters have been added that indicate which child models are required to be retrieved from the published EDIF zip files.

5. You can also edit the component's pins from this dialog by clicking on the **Edit Pins** button to display the *Component Pin Editor* dialog, or double-click on a Pin name in the **Pins** section of the *SCH Library* panel to display the *Pin Properties* dialog. Rearrange the pins as required and save the component.
6. Save the schematic library and close.

Using the Core Component

The core component can now be used in a schematic. Make sure you remove the appropriate schematic file from the project if you are using the core component to replace an existing section of an FPGA design.

As mentioned earlier, when you synthesize a design that uses the new core component, the system will search the user EDIF folders for the required EDIF models. The location of the user EDIF models is specified in the *FPGA – Synthesis* page of the *Preferences* dialog. The search sequence for EDIF models is:

```
$project_dir
$user_edif\$vendor\$family
$user_edif\$vendor
$user_edif
$system_edif\$vendor\$family
$system_edif\$vendor
$system_edif.
```

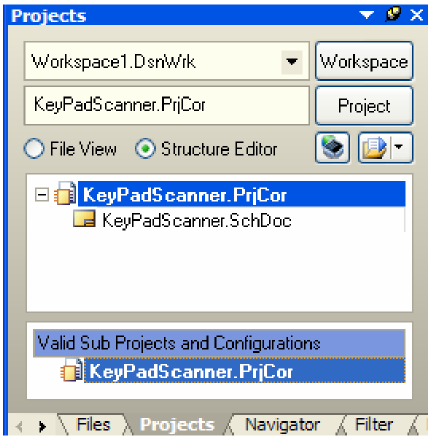
Note that the search locations include the project directory, useful if you need to transfer the design to another PC that does not have the user EDIF models location defined.

Linking to the Core Project during Development

If you are still developing the core while using it in an FPGA project, you can link from the core component symbol to the core project. If this is done, when you synthesize the FPGA project, the system will search in the `Project Outputs` folder of the core project for the required EDIF models, rather that retrieving it from the user EDIF folders. Note that linking between a component and the core project is done by name as well, so the component has to be named the same as the project.

To link from the component symbol in the FPGA project to the core project that you used to create the component:

1. Go to the *Projects* panel and click on **Structure Editor** button to display the project structure view. Make sure that your project is compiled (**Project » Compile Core Project**) so that it appears in the structure tree. You will see the **Valid Sub Projects and Configurations** list in the bottom section of the *Projects* panel.



2. Select the software project that we just created, `KeyPadScanner.PrjCor`, from the Valid Sub-Projects list by clicking on its icon and drag-and-drop it onto the icon of the processor component KeyPadScanner in the top section of the panel. Note that valid targets will be highlighted. The link will be established and the structure will recompile to re-establish the integrity.

Revision History

Date	Version No.	Revision
21-Jan-2004	1.0	New product release
14-Jan-2005	1.1	Updated for SP2. Changes to Project Options, FPGA Preferences, Configuration Manager and Symbol Options dialogs.
07-Jul-2005	1.2	Updated for Altium Designer SP4.
12-Dec-2005	1.3	Path references updated for Altium Designer 6
31-Oct-2006	1.4	Path references, text formatting and images updated.
04-Mar-2008	2.0	Updated for Altium Designer Summer 08