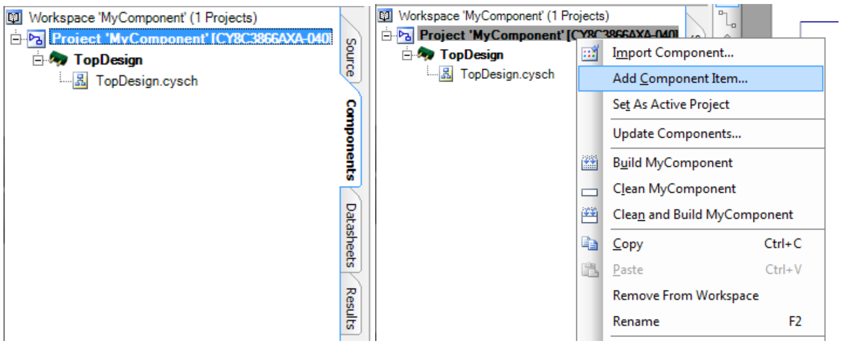
Creating A Verilog-Based Component

Contents

Generating the Verilog File

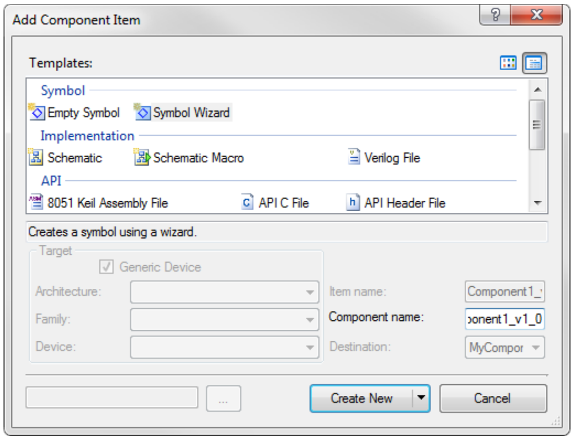
1. Open **PSoC® Creator™**.
2. Create a new project or open an existing project (for any PSoC device) This guide will use the project name “*MyComponent*”.
3. Click on the ‘**Components**’ tab of the **Workspace Explorer**.

   Right-Click on Project, and select **Add Component Item** from the context menu.



4. Select **Symbol Wizard** and change the component name to what you want – this guide uses Component1_v1_0 as the component name. Click **Create New**.

PSoC Creator supports versioning by adding _vX_Y to the Component name; where X is the major version, and Y is the minor version.



Creates a symbol using a wizard.

Target

☑ Generic Device

Architecture:

Family:

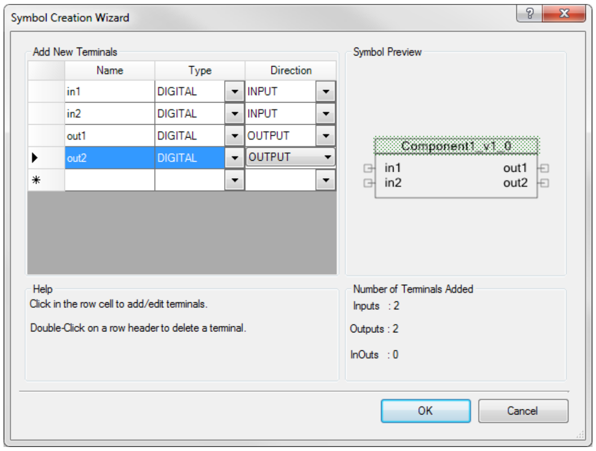
Device:

Item name: Component1_

Component name: ›onent1_v1_0

Destination: MyCompor

5. Add terminals to the component as required. You can always add more later. Click **OK**.

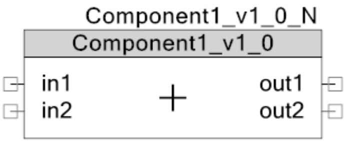


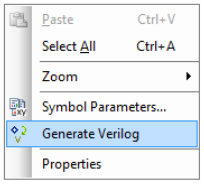
Name	Type	Direction
in1	DIGITAL	INPUT
in2	DIGITAL	INPUT
out1	DIGITAL	OUTPUT
out2	DIGITAL	OUTPUT

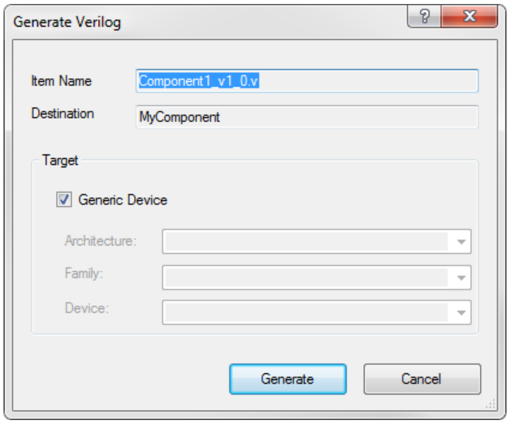
Help
Click in the row cell to add/edit terminals.
Double-Click on a row header to delete a terminal.

Number of Terminals Added
Inputs : 2
Outputs : 2
InOuts : 0

6. Right-click on a blank space in the symbol file. Select **Generate Verilog** from the context menu. Click **Generate** on the window that appears.







Item Name: Component1_v1_0.v

Destination: MyComponent

Target

☑ Generic Device

Architecture:

Family:

Device:

7. Enter the code for your Component anywhere between the `` `#start `` and `` `#end `` comments in the file.

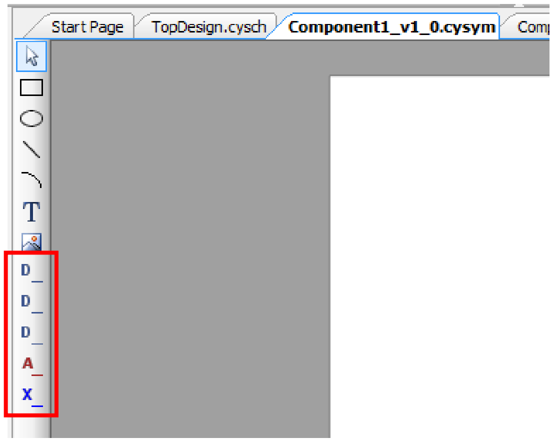
Regenerating Verilog

8. After generating the Verilog code for the first time, if you add a terminal or parameter, you may want to regenerate the Verilog to bring the newly added parameter or terminal to the Verilog file. You can regenerate the Verilog by right-clicking on an empty space on the schematic, and selecting **Generate Verilog**. However, you will lose all Verilog code written outside the `` `#start `` and `` `#end `` comments.

   You will never need to regenerate the Verilog, because you can manually add the parameter or terminal name to the Verilog code.

Adding Terminals

9. In the symbol file, either use hotkeys (I, O, B) or the buttons shown in the following figure to add terminals to the symbol.

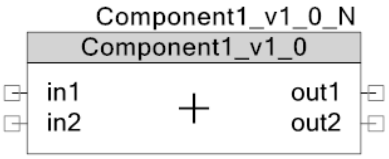


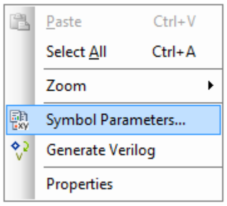
10. For these terminals to show up in the Verilog file, generate the Verilog file again. Alternatively, you can manually add these terminals as inputs or outputs to the module port list.

Adding Parameters

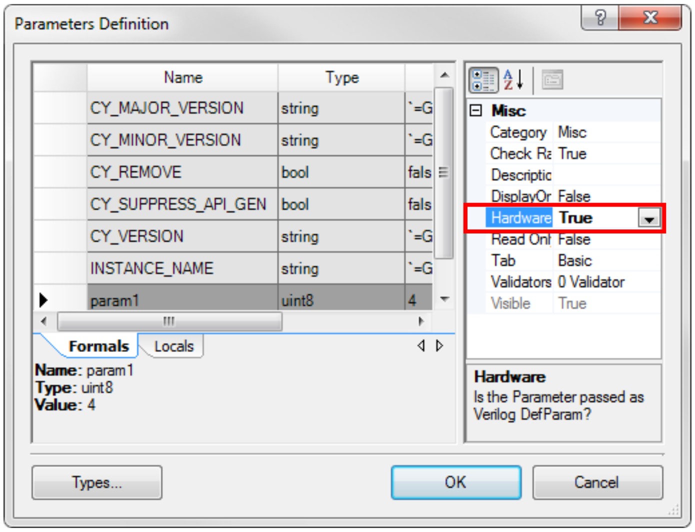
Parameters are used to configure the component from the Component customizer.

11. Right-click on an empty space in the symbol file. Select **Symbol Parameters** from the context menu.





12. Add the parameter *name*, *type*, initial *value*. If you will use this parameter in the Verilog, set the **Hardware** to **True**. You can also add the parameter **Description**, **Validator**, and so on in this window. For more information on parameters and validators, see *Component Author Guide* in PSoC Creator **Help** > **Documentation.**



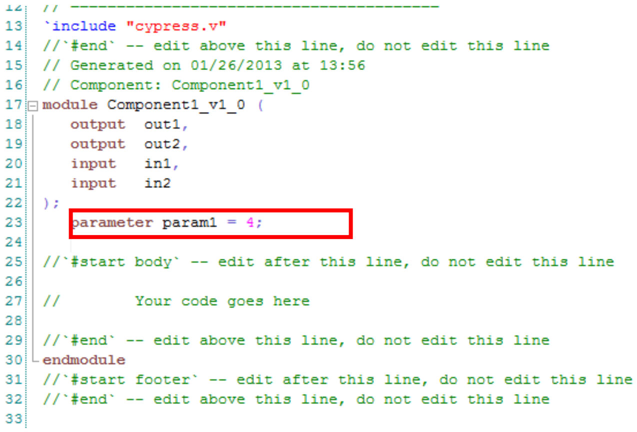
Name	Type	
CY_MAJOR_VERSION	string	`=G
CY_MINOR_VERSION	string	`=G
CY_REMOVE	bool	fals
CY_SUPPRESS_API_GEN	bool	fals
CY_VERSION	string	`=G
INSTANCE_NAME	string	`=G
param1	uint8	4

Name: param1
Type: uint8
Value: 4

Misc	
Category	Misc
Check Ra	True
Descriptic	
DisplayOr	False
Hardware	**True**
Read Onl	False
Tab	Basic
Validators	0 Validator
Visible	True

Hardware
Is the Parameter passed as Verilog DefParam?

13. Regenerate the Verilog code. The parameter appears in the Verilog code if you have set **Hardware** to **True**.



```
// ----------------------------------------
`include "cypress.v"
//`#end` -- edit above this line, do not edit this line
// Generated on 01/26/2013 at 13:56
// Component: Component1_v1_0
module Component1_v1_0 (
    output  out1,
    output  out2,
    input   in1,
    input   in2
);
    parameter param1 = 4;

//`#start body` -- edit after this line, do not edit this line

//        Your code goes here

//`#end` -- edit above this line, do not edit this line
endmodule
//`#start footer` -- edit after this line, do not edit this line
//`#end` -- edit above this line, do not edit this line
```

Adding A Datapath Instance

14. Open the Datapath configuration tool from **Tools**>**Datapath Config Tool**.

15. Open the Verilog file that you have created in the Datapath Config Tool from **File**>**Open**.

16. Add a new datapath instance to this Verilog file by choosing **Edit**>**New Datapath** and select a Datapath instance type from the generic, 8, 16, 24 and 32-bit Datapath instance types.

17. When you complete your changes in the Datapath Config Tool, save the file (**File**>**Save**) before you switch to the Verilog file. Similarly, save your changes in the Verilog file before returning to the Datapath Configuration Tool.