



Efinity IP Packager User Guide

UG-EFN-IP-PACKAGER-v1.2
November 2024
www.efinixinc.com



Contents

Introduction.....	3
Open the IP Packager.....	4
Packaging New IP Using the RTL Scanner.....	5
Import an Existing Configuration.....	8
Save.....	8
Setting Up the IP.....	9
General Tab.....	10
Device Blacklist Tab.....	11
FileSet Tab.....	13
Choice Tab.....	18
Parameters Tab.....	20
Custom Rule Checker.....	23
Ports Tab.....	26
GUI Configuration Tab.....	28
Review and Package Tab.....	31
Import Self Packaged IP into IP Manager.....	33
Expressions.....	34
Type Literal.....	34
Arithmetic Operator.....	35
Relational Operator.....	35
Logical Operator.....	35
Reference Operator.....	36
Conditional Operator.....	36
Supported Function Calls.....	37
Using Custom VHDL Libraries.....	38
Revision History.....	42

Introduction

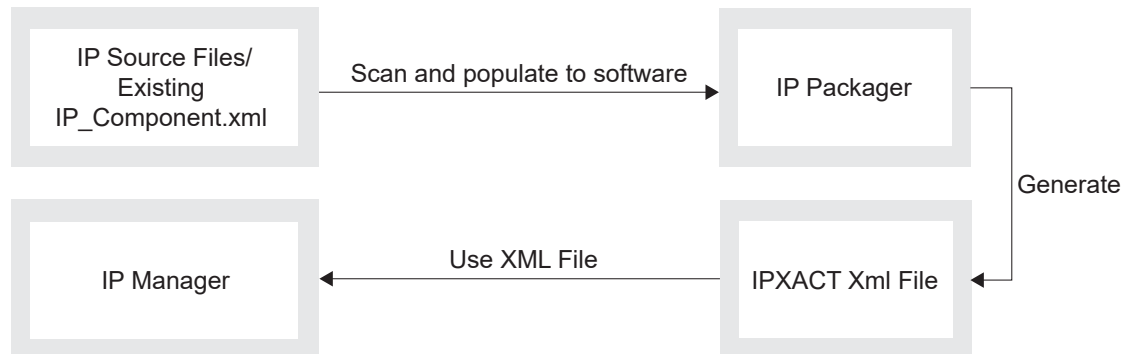
The IP Packager is a tool that lets you "package" design file(s) as standalone IP for use in the IP Manager. This feature is helpful for using the same code in multiple projects.



Note: The IP Packager is available in the Efinity software v2024.1 and higher.

The following figure shows the flow of IP Packager in relation to IP Manager.

Figure 1: IP Packager Flow



Note: You can also use the IP Packager as a standalone tool, independently from the IP Manager.

Open the IP Packager

To open the IP Packager:

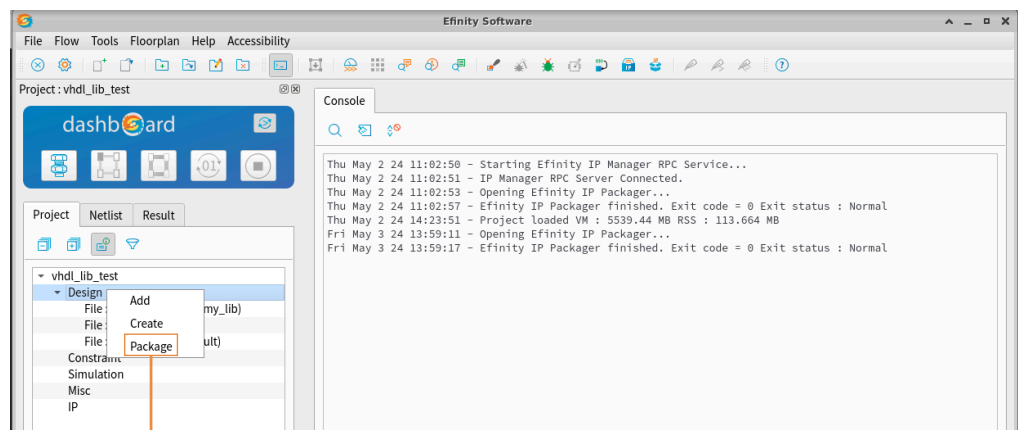
- Click the IP Packager icon located on the Efinity toolbar.

Figure 2: Open IP Packager from the Efinity Toolbar



- Right-click **Project tab** > **Design** and select **Package**.

Figure 3: Open IP Packager from the Project Tab



Select Package to
enter IP Packaging
page

Packaging New IP Using the RTL Scanner



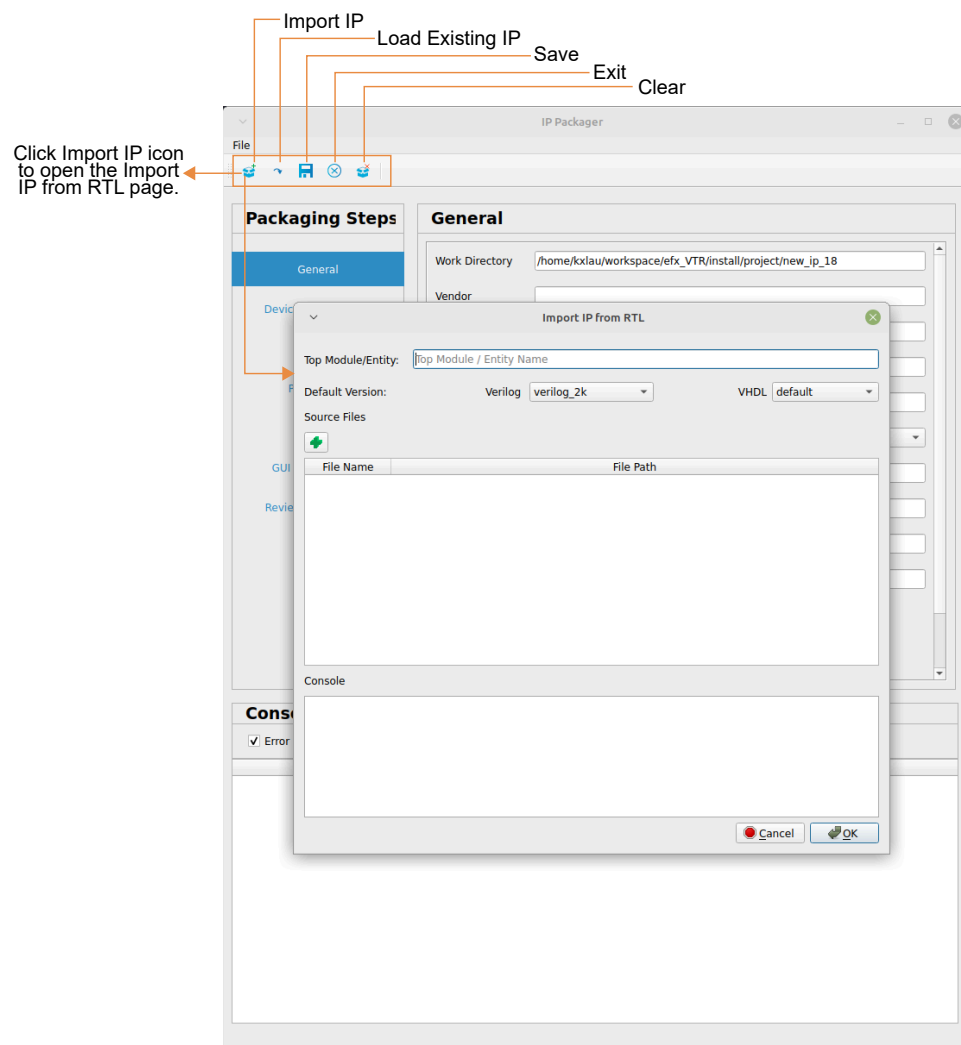
Note: Efinix recommends you to include your design files in Efinix and run **Synthesis** before performing this step to ensure that you can compile your design before packaging your design into IP. To open IP Packager, right-click on the **Design tab > Package**.

The RTL scanner helps to fill out the information that can be extracted from the RTL. To have the best mix language boundary crossing compatibility, the IP Packager supports the use of basic type declarations for port declarations.

```
Verilog: wire, reg
SystemVerilog: wire, reg, logic
VHDL: std_logic, std_logic_vector
```

The IP Packager can scan selected RTL files to package. Choose **File > Import** or click the Import IP toolbar icon. Do not use this method for IP that has already been packaged.

Figure 4: Import New IP from a Project



To import new project files:

1. Enter the name of the top module or entity.
2. Choose the default versions of Verilog HDL or VHDL.
3. Click the green plus icon to open the **Source files** dialog box.
4. Choose the files to import; you can shift-click to select multiple files.
5. Click **Open** to add the files.
6. Click **OK**. The tool analyzes and elaborates the scanned files; it does not perform synthesis.

Figure 5: Adding Source Files

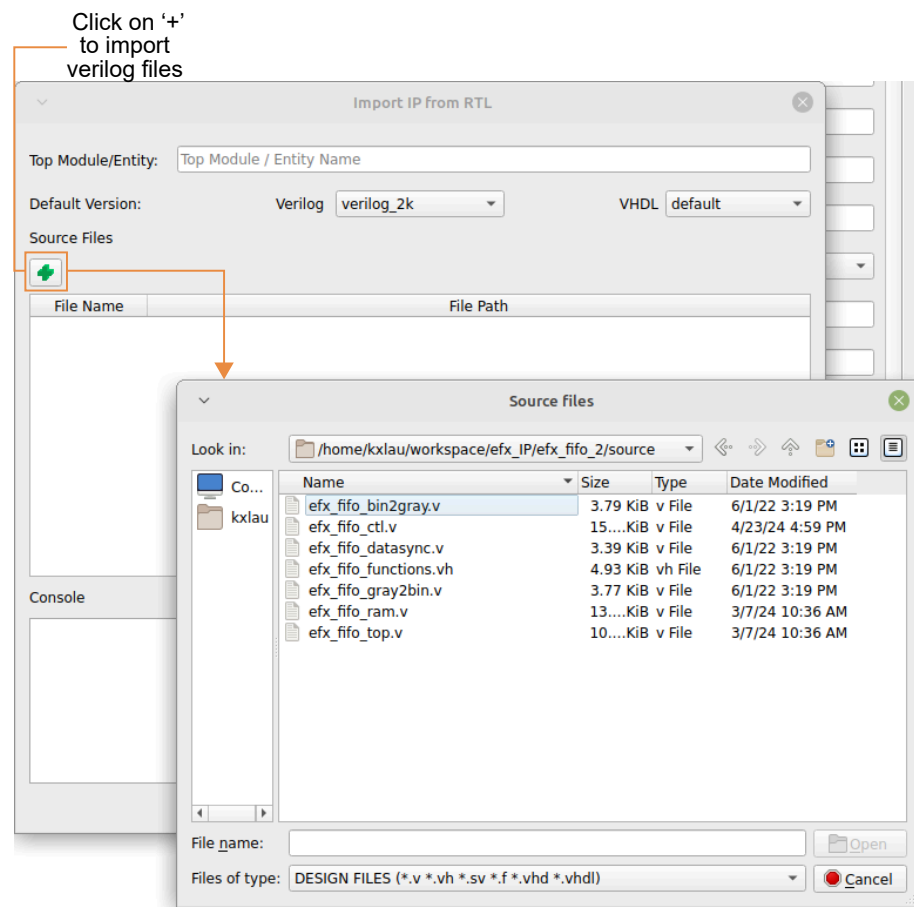
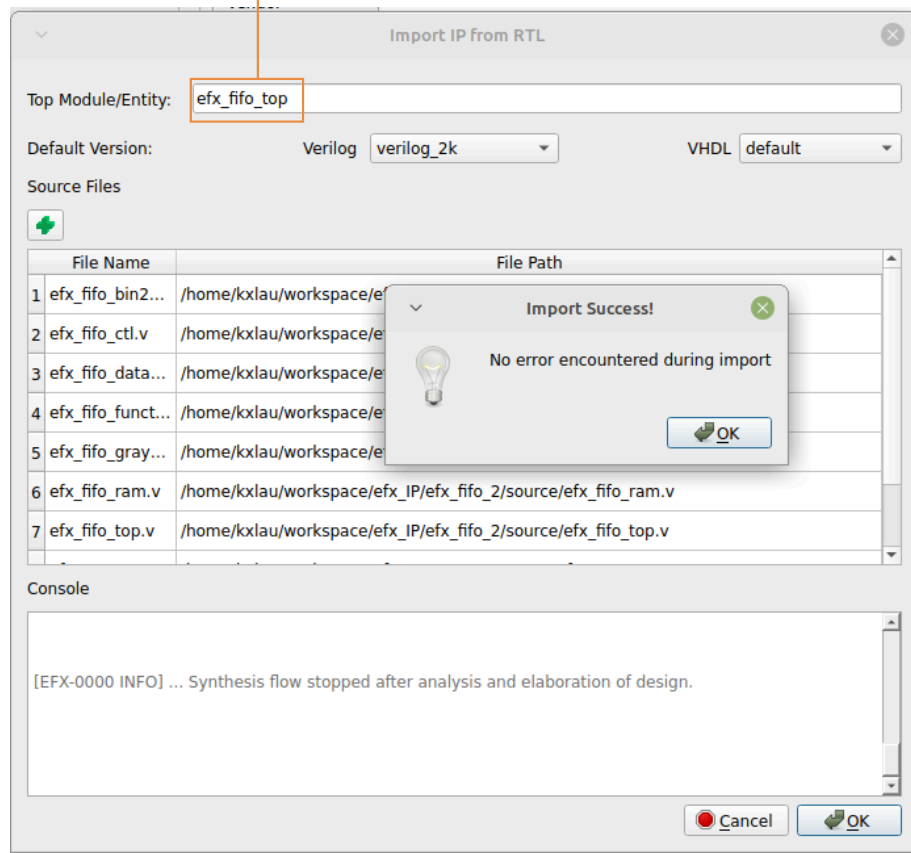


Figure 6: Import New IP Success

Enter the name for the
Top Module/Entity



Note:

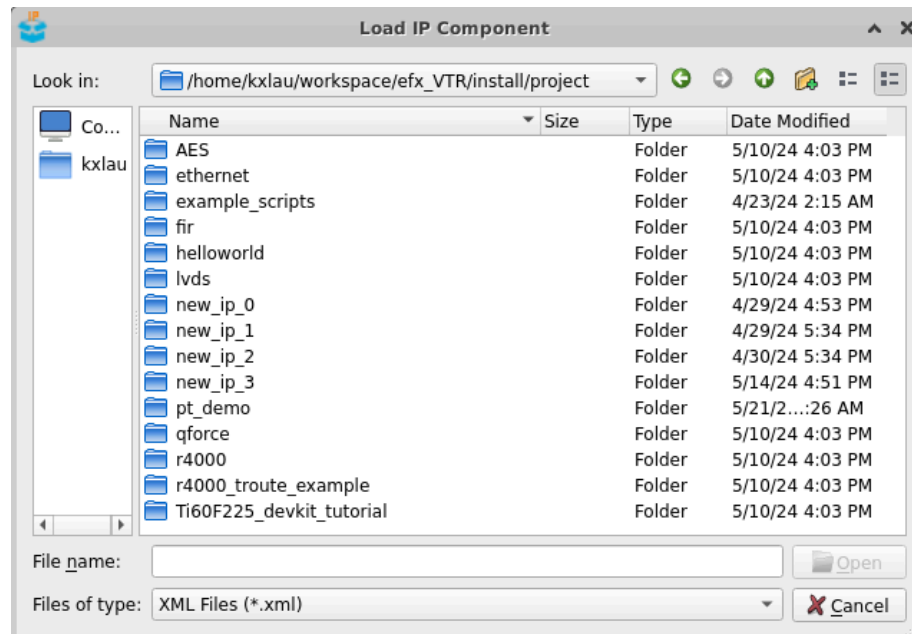
- During the import process, the IP Packager scans the RTL files for parameters and ports, eventually populates them into the corresponding tabs in the IP Packager.
- After a successful import of new IP, you need to continue to [Setting Up the IP](#) on page 9.

Import an Existing Configuration

You can import an existing configuration using the **ip_component.xml** and **config_template.json** files. These files should be located in the same working directory. To import:

1. Choose **File > Load Existing IP**. The **Load IP Component** dialog box opens.
2. Browse to your working directory and select the **ip_component.xml** file.
3. Click **Open**. The IP Packager loads the existing configuration.

Figure 7: Importing Existing Configuration



Note: The IP packager saves temporary files with the extension **.tmp**. You can also load the **ip_component.xml.tmp** so you can finish any uncompleted work you saved previously.

Save

The IP Packager has a save button that conveniently saves any work without going through all the error checking. The saved version is named **ip_component.xml.tmp** and **config_template.json.tmp**.



Note:

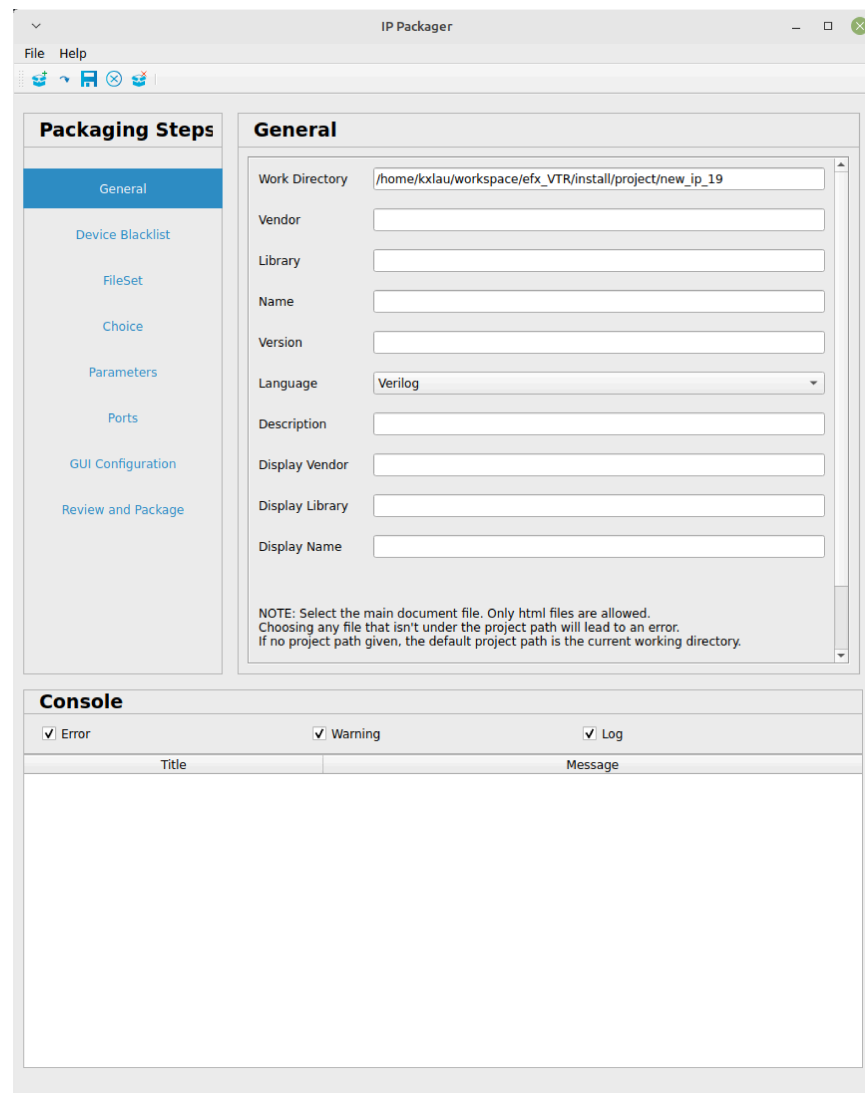
- You can load the **.tmp** files back into the IP Packager to continue the work where you left off. To use the IP inside of the IP Manager, you must package and archive the IP first.
- If you want a file dialog parameter, you have to add "" as the default value. You might encounter an error in IP Manager under the file dialog parameter that implies that the file path cannot be empty.

Setting Up the IP

When you first open the IP Packager, the only data it captures is the **Work Directory**, which points to **\$Efinity_Home/project** with `new_ip_<number>` as the IP name. The tool increments `<number>` if a project with the same name exists already. You can change the path and name to suit your project.

The left side of the IP Packager window contains links for the various pages. Click through the pages to complete the required project details.

Figure 8: IP Packager



Tip: Many of these settings are displayed in tables. Right-click on a table cell to open a context-sensitive menu to add, edit, or delete values.

General Tab

In the General tab, you set the required fields (**Vendor**, **Library**, **Name**, and **Version**) and the optional fields (display attributes and the document path).



Note: The document field is optional and uses the current working directory as the document path (if left empty).

The supported version format is $x.x$, $x.x.x$, where x is a number.



Note: If you have previously opened a packaged IP, the working directory is disabled.

Figure 9: General Tab

General

Work Directory	<input type="text" value="/home/kxlau/workspace/efx_VTR/install/ipm/ip/efx_fifo_2"/>
Vendor	<input type="text" value="efinixinc.com"/>
Library	<input type="text" value="memory"/>
Name	<input type="text" value="efx_fifo_top"/>
Version	<input type="text" value="6.0"/>
Language	<input type="text" value="Verilog"/>
Description	<input type="text" value="core provides an optimized solution using the block RAM in Trion® FPGAs."/>
Display Vendor	<input type="text" value="Efinix"/>
Display Library	<input type="text" value="Memory"/>
Display Name	<input type="text" value="FIFO"/>

NOTE: Select the main document file. Only html files are allowed.
 Choosing any file that isn't under the project path will lead to an error.
 If no project path given, the default project path is the current working directory.

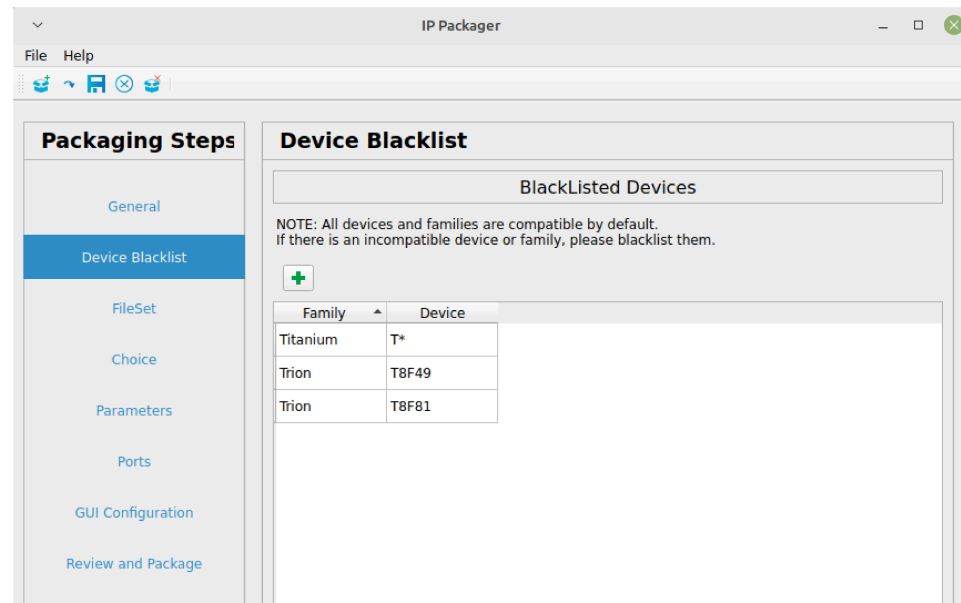
Device Blacklist Tab

The Device Blacklist tab has the list of FPGAs and packages that your IP does not support. These settings affect which FPGA's are shown in the IP Manager. If the FPGA is listed on the Device Blacklist tab, you cannot see the IP in the IP Catalog if your project is using a blacklisted device.



Note: Right-click a table cell to open a pop-up menu with **Edit** and **Delete** options..

Figure 10: Device Blacklist Tab

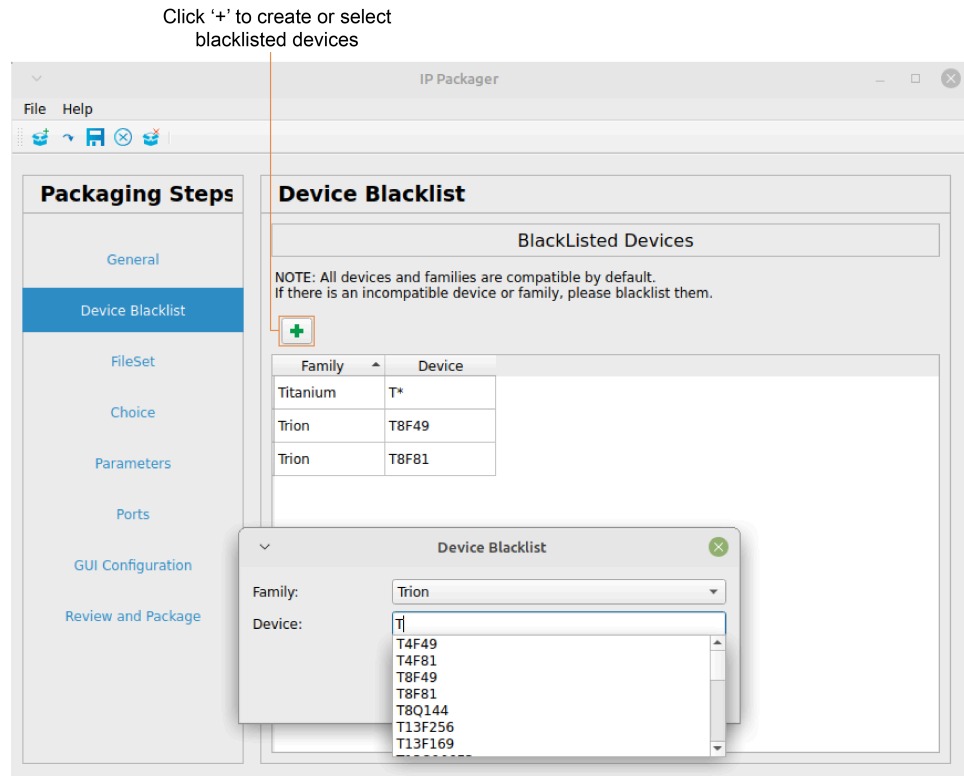


To add an FPGA to the blacklist:

1. Click the plus icon to open the Device Blacklist dialog box.
2. Select Trion or Titanium from the **Family** drop-down list box.

3. Start typing the FPGA name in the **Device** box to display a picklist.

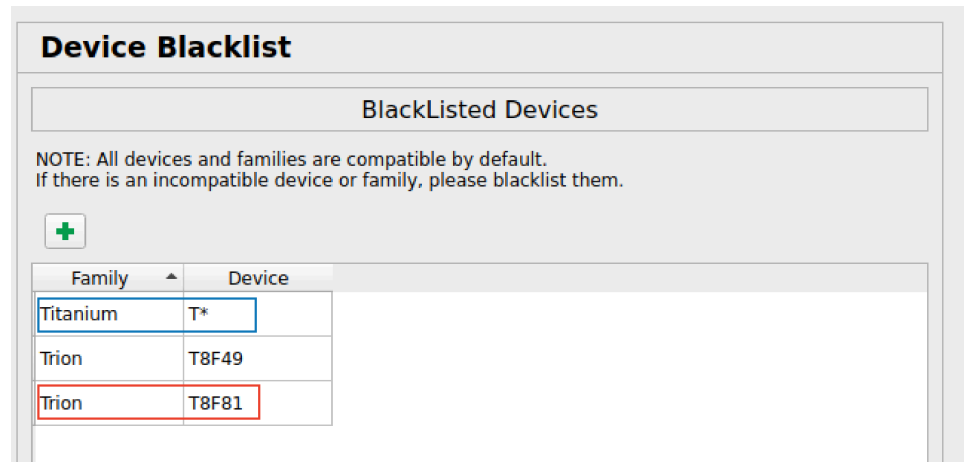
Figure 11: Add a Device to the Blacklist



If you want to blacklist an entire selection of devices, you can use a glob pattern function in the **Device** box (e.g., T*). As shown in the following figure,

- The red box is the device that is selected from the dropdown.
- The blue box has all Titanium devices blacklisted.

Figure 12: Blacklisted Devices



FileSet Tab

In this tab you list all of your IP RTL and associated files. You organize your files in a *fileset*; each fileset is a collection of one or more files. You can add multiple filesets. When you click the fileset name, the tool displays the list of files it contains in the **File Summary** box.

Tip: Right-click a table cell to open a pop-up menu with **Edit** and **Delete** options.

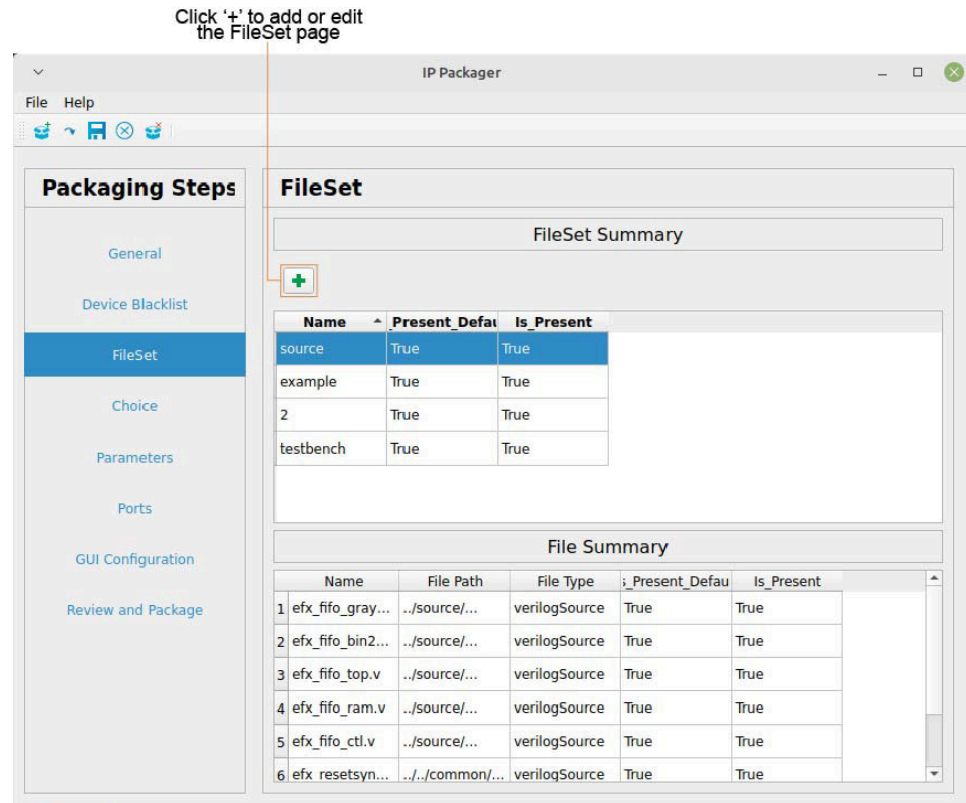
Remember: You must create a fileset with the name `source`, fileset type `external source`, and add all the included source files during the IP import. You might encounter an error in IP Manager when you generate this IP.

You can only have one fileset source in each IP because the design files under the fileset source are concatenated into an IP wrapper file during the IP generation in IP Manager. If there is no fileset, then the software issues an error message.

Table 1: Edit IP Fileset Options

Option	Description
Name	The folder name created in your working directory.
Display Name	The name that is shown on the deliverable page in IP Manager.
Type	<p>There are 4 types of filesets category.</p> <ul style="list-style-type: none"> • External Source: For source design files. You should include all the files you put in the Import IP page in the IP Packager when running the <code>RTL_Scanner</code>. • External Example: All the example design files should be included into this fileset. • External Testbench: All the testbench design files should be included into this fileset. • External Script: All files in this fileset that function as a standalone third-party script.

Figure 13: Fileset Tab



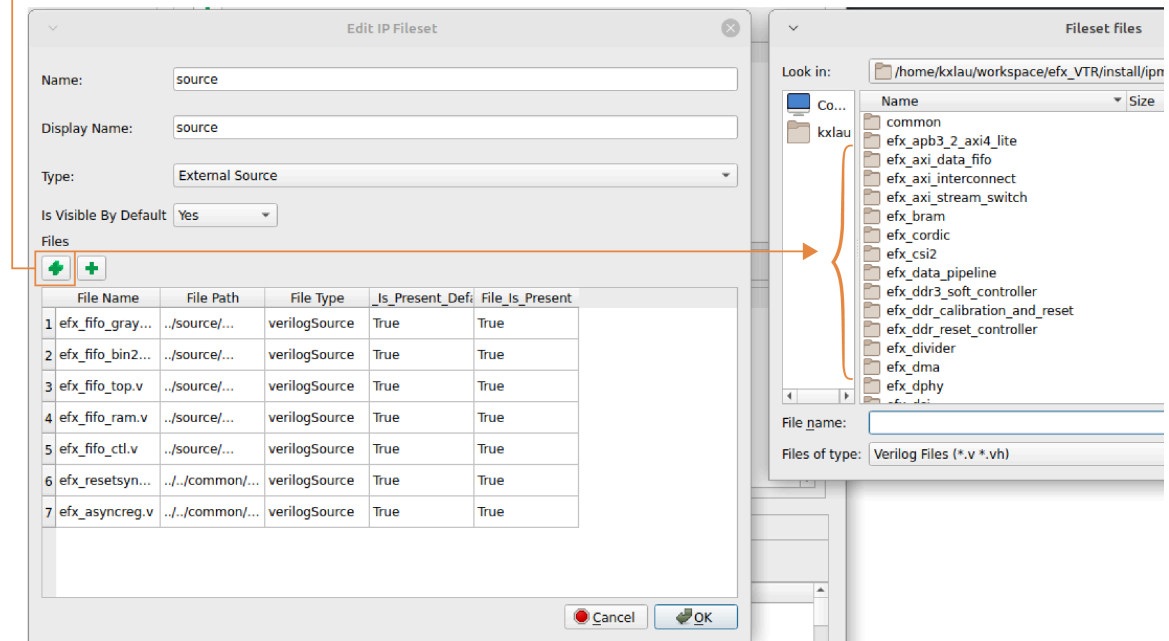
To add a fileset:

1. Click the green plus icon. The Edit IP Fileset dialog box opens.
2. Enter an internal name (**Name** box) and a name to show in the IP Catalog (**Display Name** box).
3. Choose the fileset type (**Type** drop-down list box).

4. Add multiple files:

Figure 14: Multi Select Option

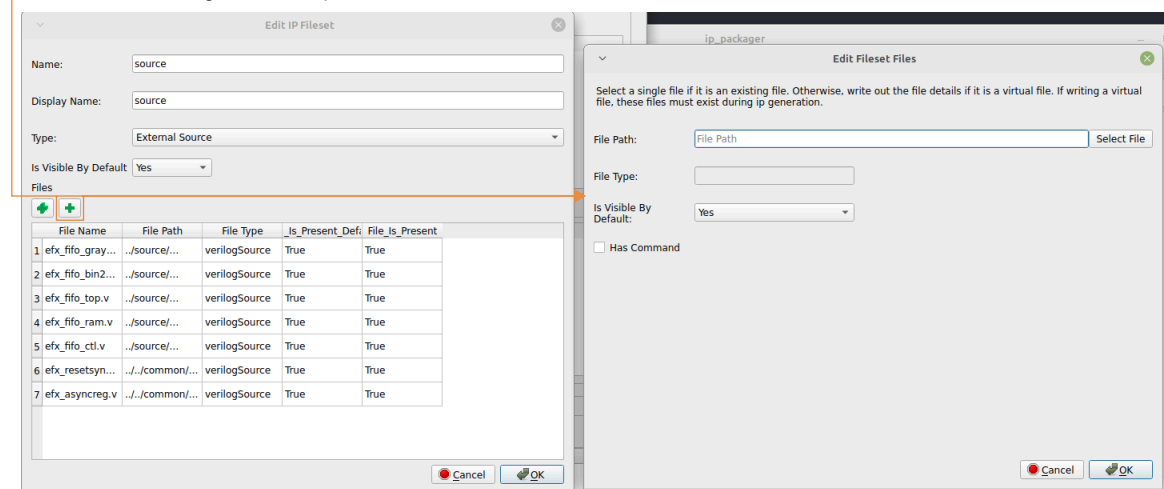
Click '+' to multi-select files. You can select the number of files needed.



- Click the bold green plus icon to add multiple files. The **Fileset files** dialog box opens.
 - Browse to a directory and shift-click to select multiple files.
 - Click **Open**. The tool adds the fileset and files.
5. Add a single file:

Figure 15: Add Single File Option

Click '+' to use single add file option.



- Click the small green plus icon to add a single file. The **Edit Fileset Files** dialog box opens.
- Click **Select file** to browse to the file you want to add.
- Click **Open** to select the file.

- d. Optionally, add a built command for the file by turning on the **Has Command** option and entering the command and flags. The default file type is Python (**.py**).

Figure 16: Build Command

Select a single file if it is an existing file. Otherwise, write out the file details if it is a virtual file. If writing a virtual file, these files must exist during ip generation.

File Path:

File Type:

Is Visible By Default:

Has Command

Build_Command

Command:

Flags:

Table 2: Build Command Options

Option	Description
Command	Path to run your script.
Flags	Arguments that must be passed into the script.

- e. Click **OK** to add the fileset.

The following is the list of supported variables that can assist you in writing your build command:

Table 3: Supported Variables in Writing Your Build Command

Variable	Description
<code>\${EFINITY_PYTHON}</code>	Refer to environment variable EFINITY_HOME/bin/python3 , platform independent.
<code>\${IP_LIBRARY_ROOT}</code>	Root IP installation path in Efinity: EFINITY_HOME/ipm/ip , platform independent.
<code>\${IP_GEN_DEVICE}</code>	Target device.
<code>\${IP_GEN_FAMILY}</code>	Target family.
<code>\${IP_OUTPUT_PATH}</code>	Output directory of the IP. Example: my_project/ip/<new_ip> , IP_OUTPUT_PATH="my_project/ip"
<code>\${IP_GEN_MODULE}</code>	Generated module name of the IP.
<code>\${IP_USER_PARAMETERS}</code>	Resolved key-value pairs parameters from the IPM.
<code>\${PROJECT_NAME}</code>	Efinity project name of the passed in Efinity path.

Choice Tab

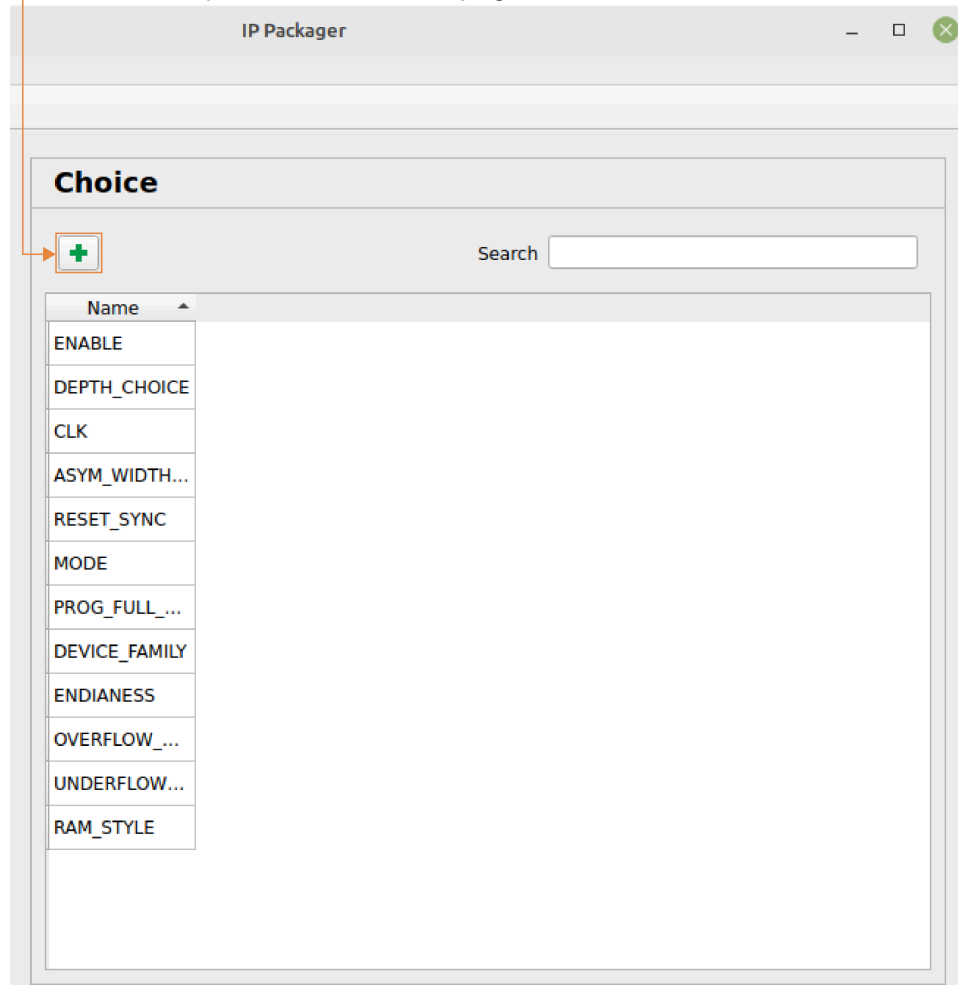
The IP Packager lets you add parameters with pre-defined choices. These settings let users configure your IP in the IP Manager. You specify the choices available to users in the **Choice** tab. You associate these choices with parameters in the **Edit Parameters** tab.



Note: Right-click a table cell to open a pop-up menu with **Edit** and **Delete** options.

Figure 17: Choice Tab

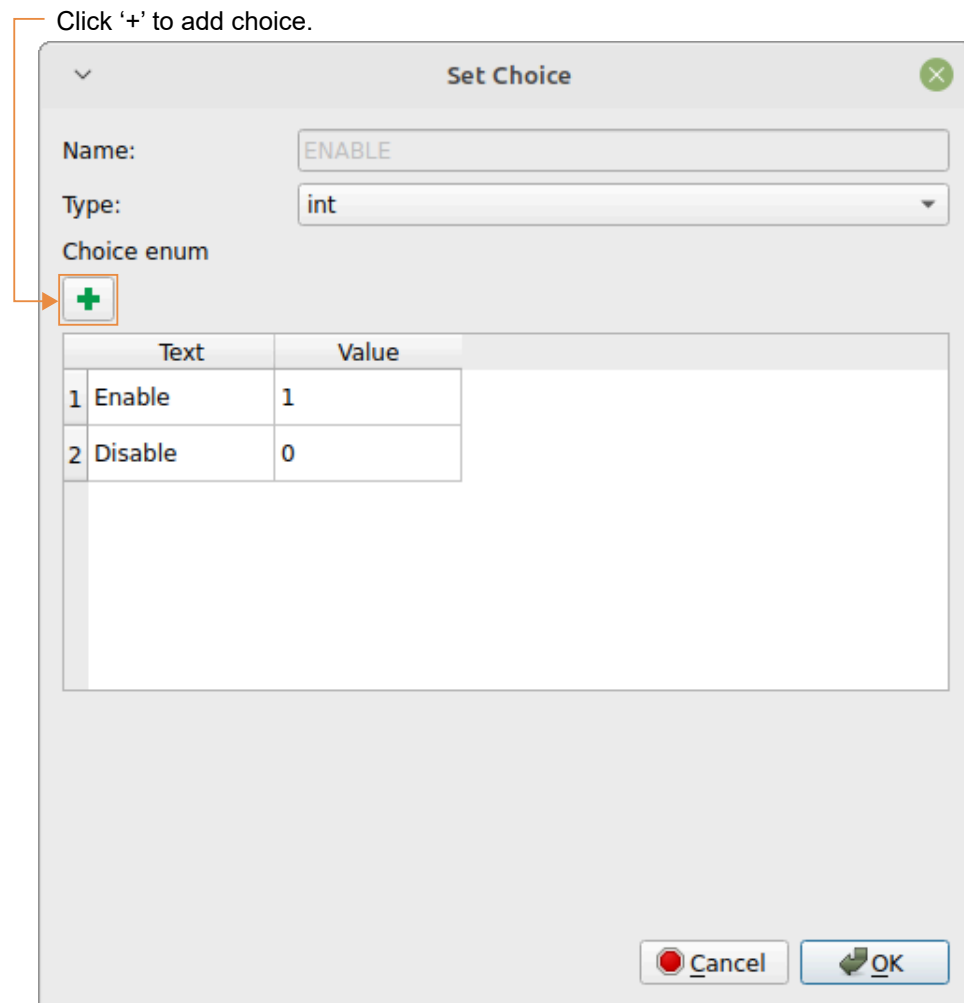
Click '+' to open the Set Choice page.



To add a choice:

1. Click the green plus icon. The **Set Choice** dialog box opens.
2. Enter the name in the **Name** field.
3. Choose the kind of value you want in the **Type** drop-down list box. The options are **int**, **bit**, **string**, **unsigned**, and **signed**.
4. Click the green plus icon. The **Choice Enum** dialog box opens
5. Enter the name (**Text**) and value (**Value**) pair for the option.
6. Click **OK** twice to save your settings..

Figure 18: Add/Edit Set Choice Page

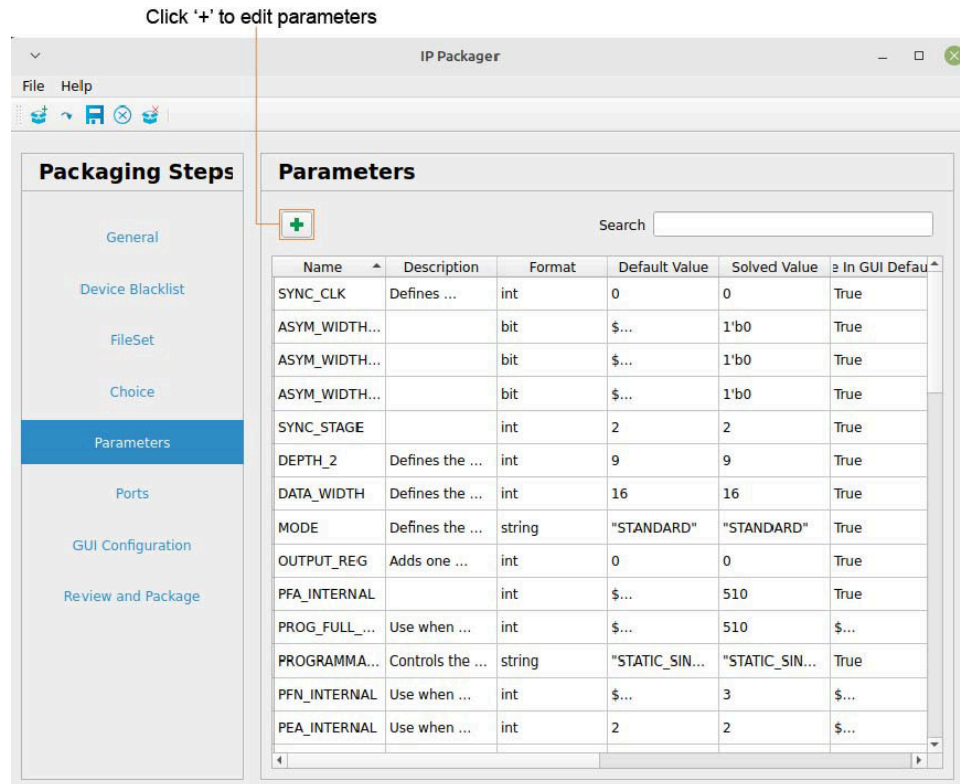


Important: Make sure that your choices match with the type you have specified. For example, if you choose `int`, you must use an integer. For `Text`, use a string and enclose it in double quotation marks (e.g., "abc").

Parameters Tab

In the **Parameters** tab you add user-configurable parameters for your IP. You use these parameters in the **GUI Configuration** tab to build the IP GUI that is displayed in the IP Manager.

Figure 19: Parameters Tab



Click the green plus icon to open the **Edit Parameter IP** dialog box.

Figure 20: Edit Parameter IP Tab

Use the options below to customize how the IP will appear in the GUI.

Name:

Specify as module parameter

Visible in GUI:

Editable:

Display Name:

Description:

Format:

Resolve:

Specify as range

Type:

Choice	Choice Enum
ENABLE	
DEPTH_CHOICE	
CLK	
ASYM_WIDTH_RATIO	
RESET_SYNC	
MODE	
PROC_FULL_EMPTY	

	Text	Value
1	STANDARD	"STANDARD"
2	FWFT	"FWFT"

Custom Rule Checker

Enable	Constraints

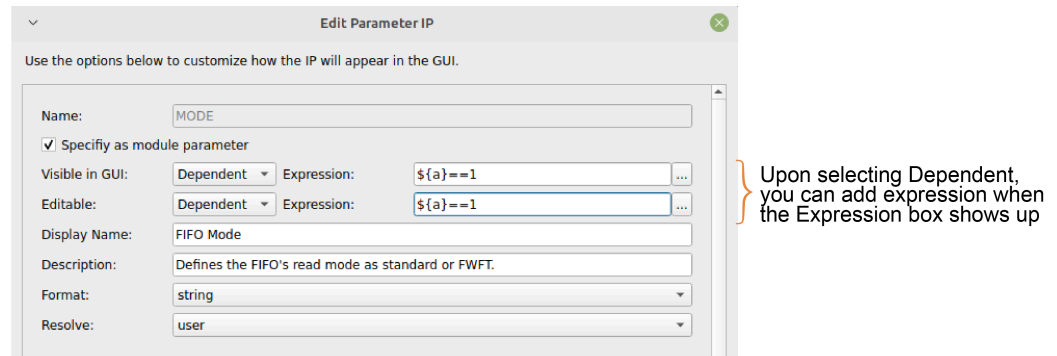
The following table describes the options you can set.

Table 4: Parameters on the Edit Parameter Page

Option	Choices	Description
Name	User defined	Specify a name. It cannot be a reserved keyword in the language the top module uses. Additionally, it cannot have any special characters other than underscore (_).
Specify as module parameter	On, off	If on, the parameter is used in the IP Manager.
Visible in GUI	Yes, No, Dependent	Indicates whether the parameter is displayed in the GUI. Yes: Shown (default). No: Hidden. Dependent: When selected, enter an expression (see Expressions on page 34).
Editable	Yes, No, Dependent	Indicates whether the parameter is user editable in the GUI. Default: Yes. Yes: Editable (default). No: No editable. Dependent: When selected, enter an expression (see Expressions on page 34).
Display Name	User defined	The parameter name shown in the IP Manager.
Description	User defined	The parameter description shown in the IP Manager.
Format	int, bit, string, unsigned, signed.	Parameter type. int: ShortInt, Int, LongInt, Real, ShortReal bit: Bit string: String unsigned: Bit signed: Bit
Resolve	user, immediate	Parameter visibility. user: The parameter is visible and user editable in the IP Manager. immediate: The parameter is not visible and not editable in IP Manager by default. However, you can still show this parameter in IP Manager if you add the parameter in the GUI Configuration tab manually.
Specify as Range	On, off	Turn on to use a range for the parameter values.
Type	List of Choices, Min Max Range	This option is available when you turn on Specify as Range . List of Choices: The parameter shows as a drop-down list box in the IP Manager. Click the choice name to use in the Choice column. The list of choices is shown for reference. Note: The default value (in the Default Value box) must be one of the values shown under Choice Enum . Min-Max Range: The parameter shows as a numerical input field with up/down arrows in the IP Manager. Set a minimum and maximum value. If you do not specify them, the option is an entry box in IP Manager.
Custom Rule Checker	See Custom Rule Checker on page 23	Add custom rule checking.
Default Value	User defined	The value is dependent on another parameter. You can use expressions (see Expressions on page 34).

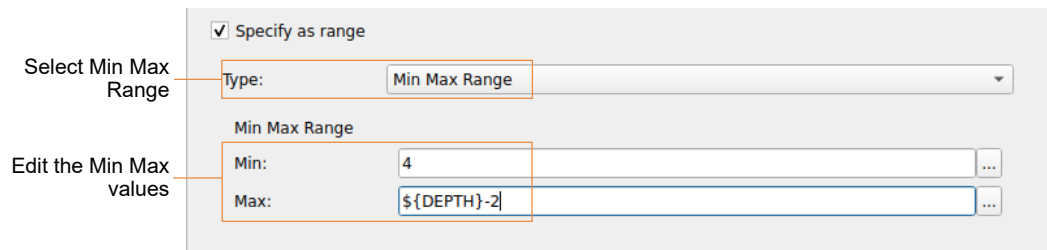
The following screen shots show these options in more detail.

Figure 21: Example of Expressions



In this example, the expression, $\$ \{ a \} == 1$ is used to check whether the value of parameter, a is equal to 1. If yes, expression is true (1'b1).

Figure 22: Setting a Min and Max Range



Custom Rule Checker

The custom rule checker feature is a powerful way to ensure that users enter the correct settings for the IP. You add *constraints* to the parameter, and the IP Manager uses those constraints to rule check the user settings. To add a new constraint, click the green plus icon. The **Edit Conditional Checker Page** dialog box opens.

Figure 23: Custom Rule Checker

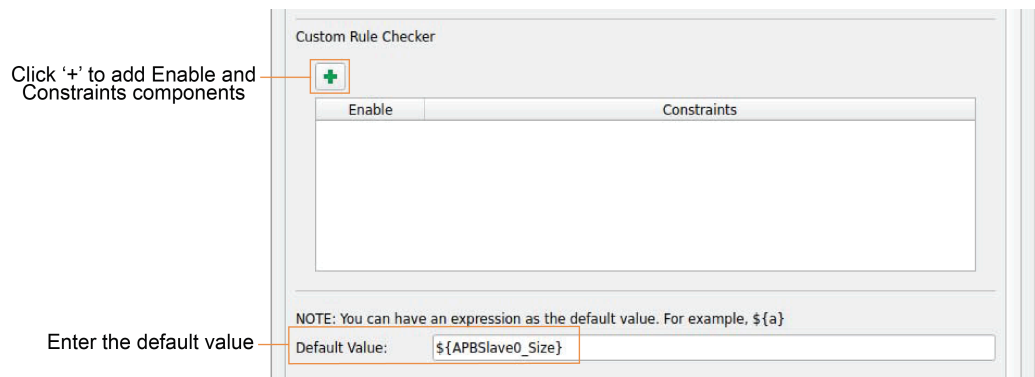
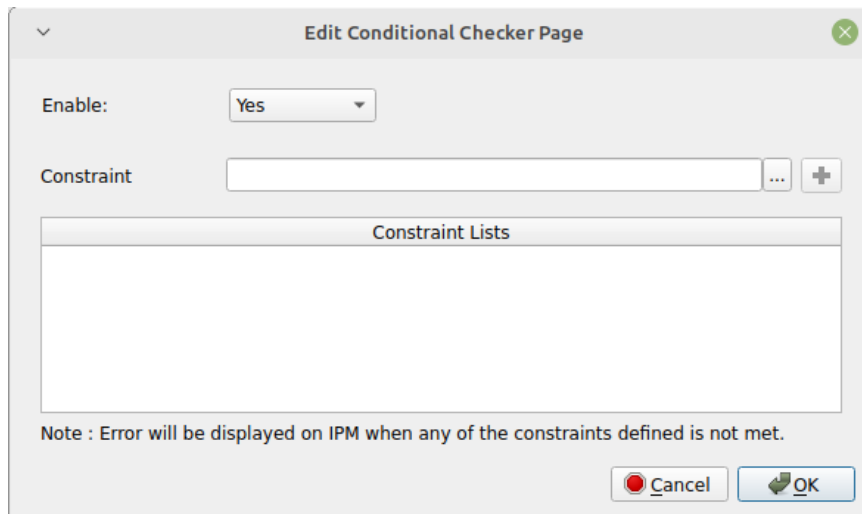


Figure 24: Edit Conditional Checker Page

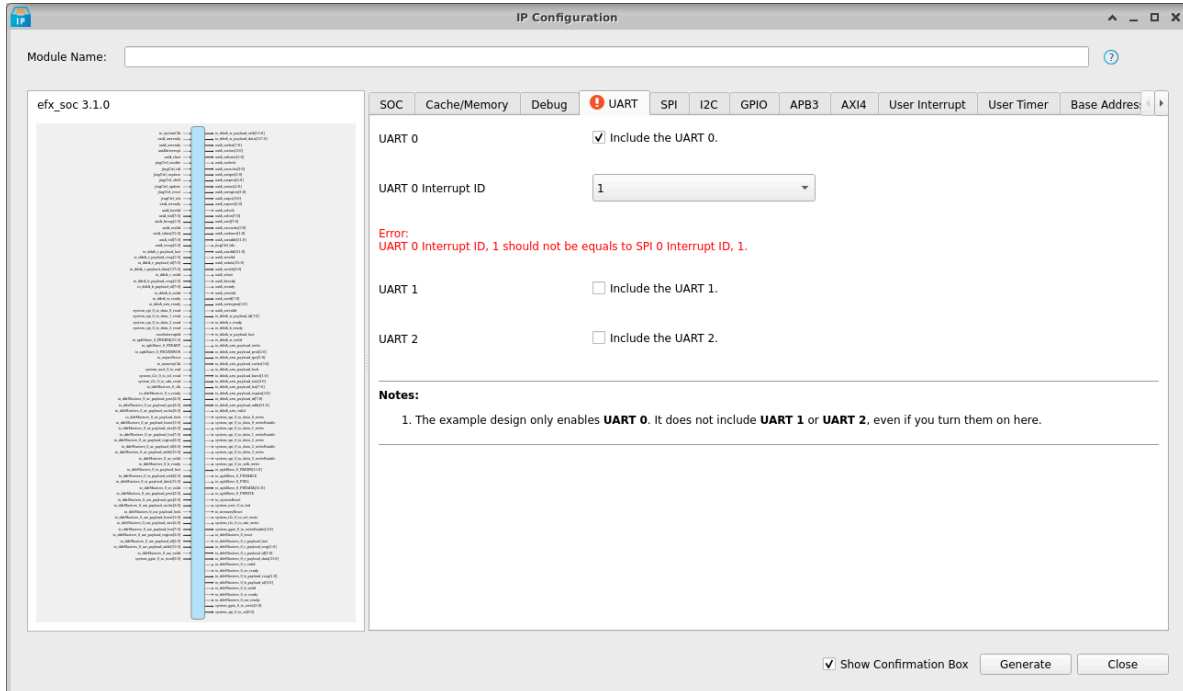


The following table describes the options.

Table 5: Enable and Constraints on Edit Conditional Checker Page

Option	Choices	Description
Enable	Yes, Dependent	<p>Yes: The rule checker is enabled by default. If the set constraint defined is not met, The IP Manager shows an error for the parameter.</p> <p>Example: The constraint is set as $\\${paramA}!\=\${paramB}$. In the IP Manager, if paramA equals paramB (suppose they are both set to a value of 1), the IP Manager shows the error "paramA, 1 should not equal to paramB, 1".</p> <p>Dependent: Enter an expression in the Expression box (see Expressions on page 34). The rule checker only applies when the expression is evaluated as true.</p>
Constraint	Expression	<p>Enter an expression and click the green plus icon. You can add multiple expressions.</p> <p>To edit or delete a constraint, right-click the expression in the Constraint Lists table and choose Edit or Delete from the pop-up menu.</p> <p>If any constraints are not met, the IP Manager shows the parameter name and an error message.</p>

Figure 25: Example Message of Constraint Not Met in IP Manager



In this example, the constraint defined is $\${UART0_INT_ID} \neq \${SPI0_INT_ID}$, which indicates that `UART0_INT_ID` cannot equal `SPI0_INT_ID`. If the value of these two IDs are equal, the IP Manager displays an error message under the parameter where the constraint is defined.

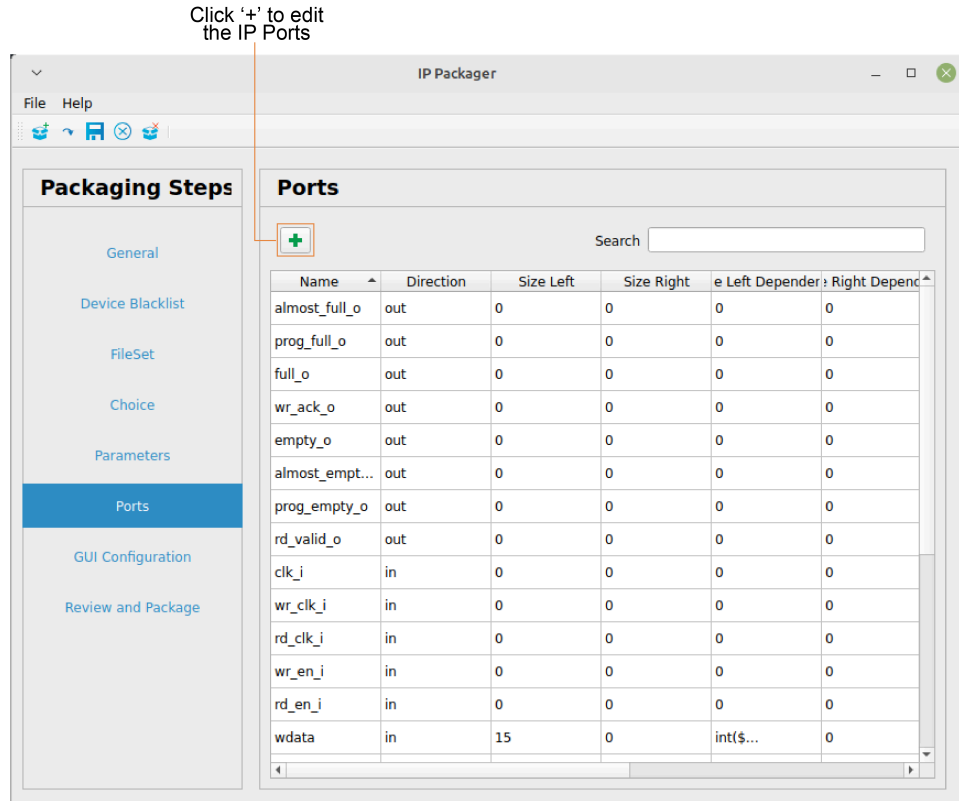


Note: The error checking only works when you click **OK** on the **Edit Parameter** page. You are notified if any error arises.

Ports Tab

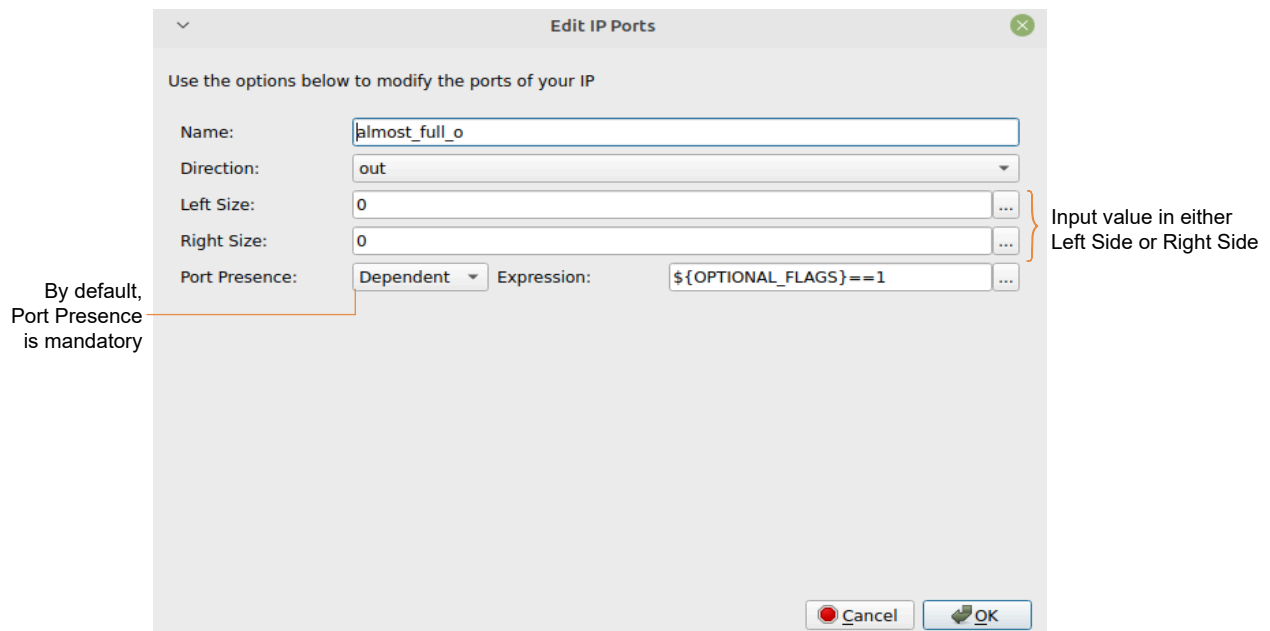
The Ports tab shows the top level input, output, and inout ports signals for your IP.

Figure 26: Ports Tab



Click the green plus icon to add a port. The **Edit IP Ports** dialog box opens.

Figure 27: Edit IP Ports Dialog Box



Enter these options:

Table 6: Edit IP Ports Options

Option	Choices	Description
Name	User defined	Enter the signal name.
Direction	in, out, inout	Choose the signal direction.
Left Size	Integer	Enter the left value.
Right Size	Integer	Enter the right value.
Port Presence	On, off	On: The port is mandatory. Off: Enter an expression (see Expressions on page 34).

GUI Configuration Tab

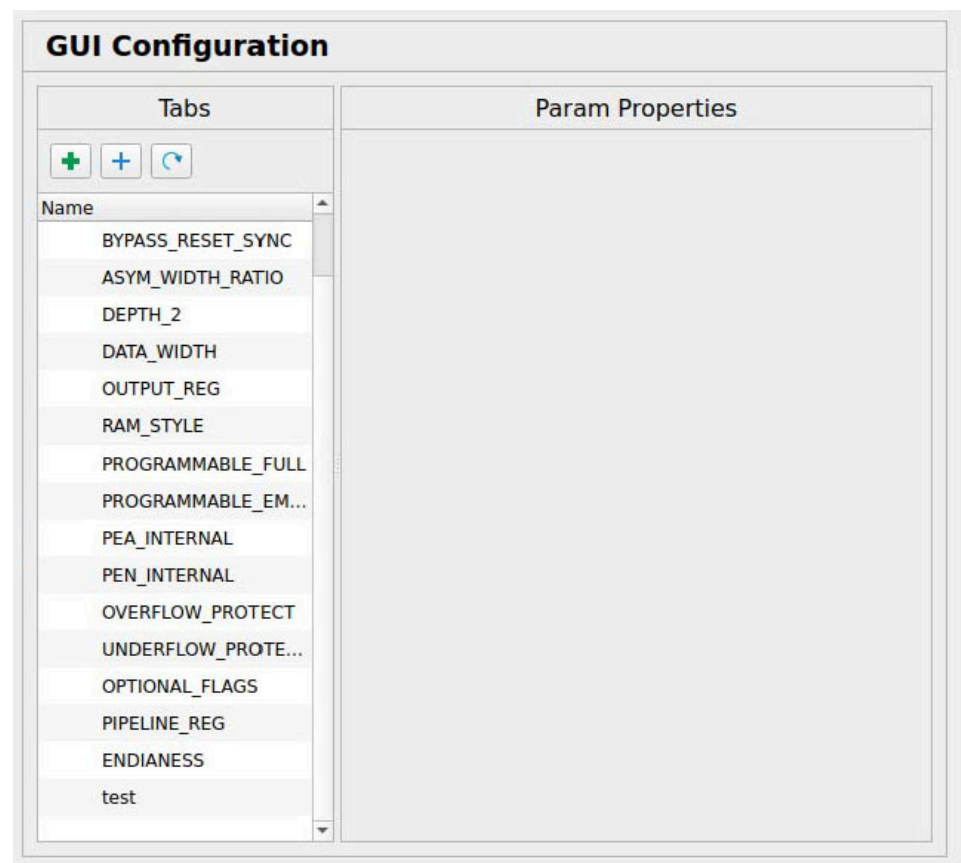
The **GUI Configuration** tab lets you customize the GUI; for example, by placing parameters on tabs that display in the IP Manager. The IP Manager displays the tabs in the same order as they appear in the **GUI Configuration** tab. First you add tabs and then add parameters to them.

- Click the bold green plus icon to add a new tab.
- Click the small green plus icon to add a parameter or note to an existing tab. Notes are additional information displayed to the user; for example, to describe the parameter or choices.
- Right-click a tab name to open a pop-up menu with **Edit** and **Delete** options.



Note: When you add a parameter with the **Resolve** option as **user**, the IP Packager automatically adds the parameter to the GUI and it displays in the **GUI Configuration** tab.

Figure 28: Automatically Added Parameters (Resolve Set to user)



Tip: You can drag and drop the parameters to re-arrange their order.

Figure 29: Add a New Tab

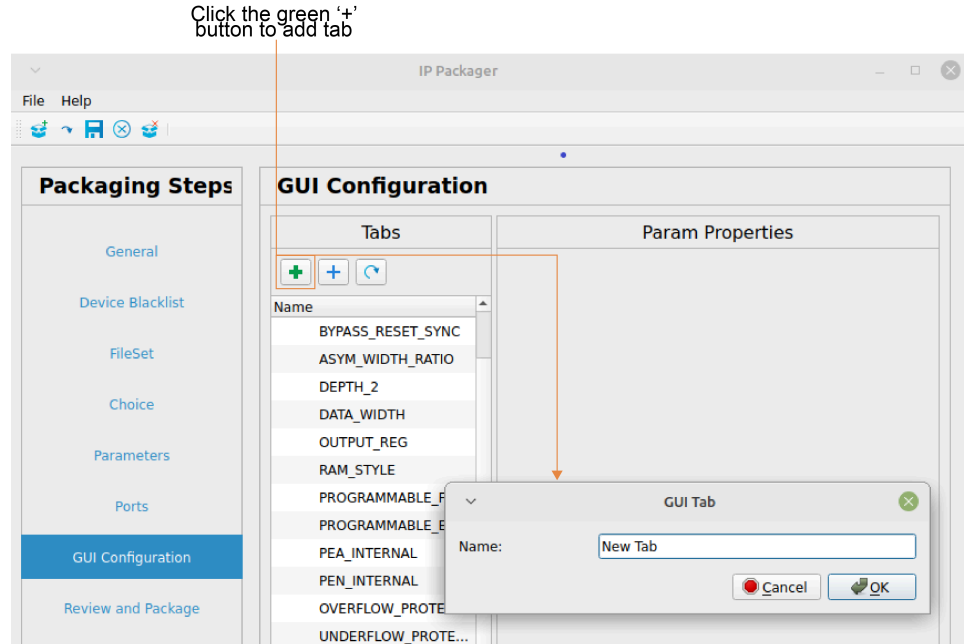


Figure 30: Add a Note

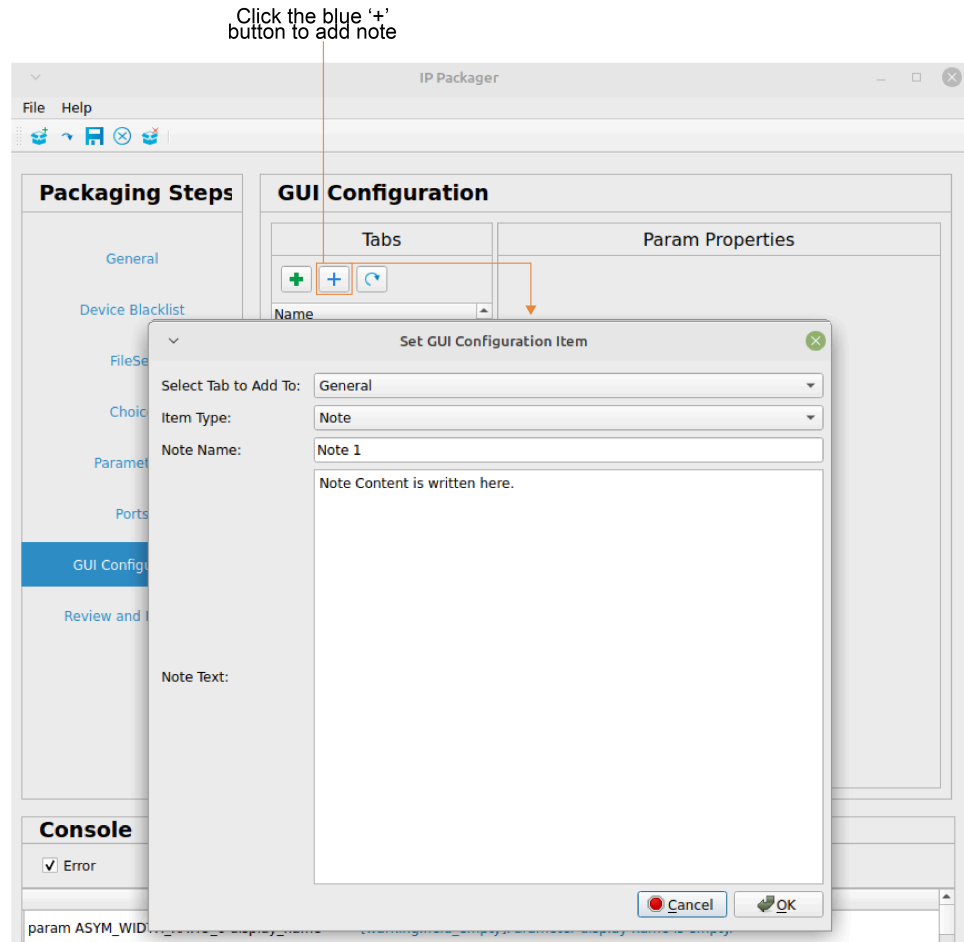
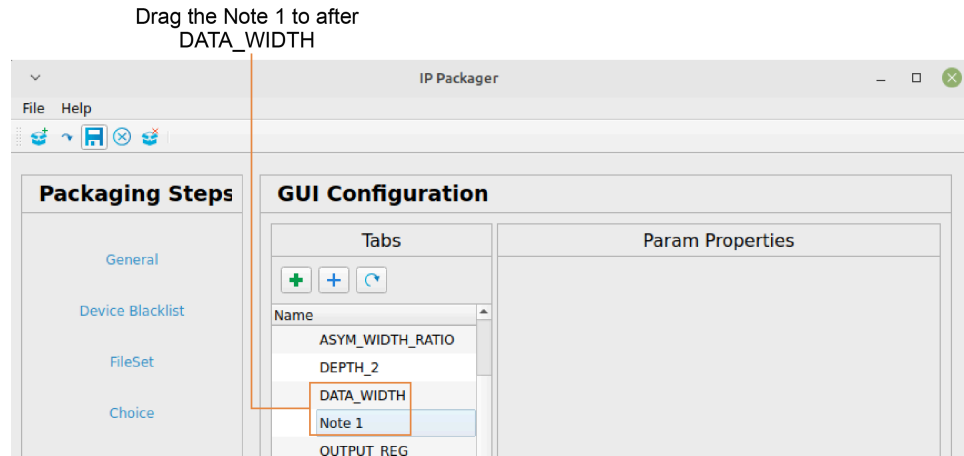


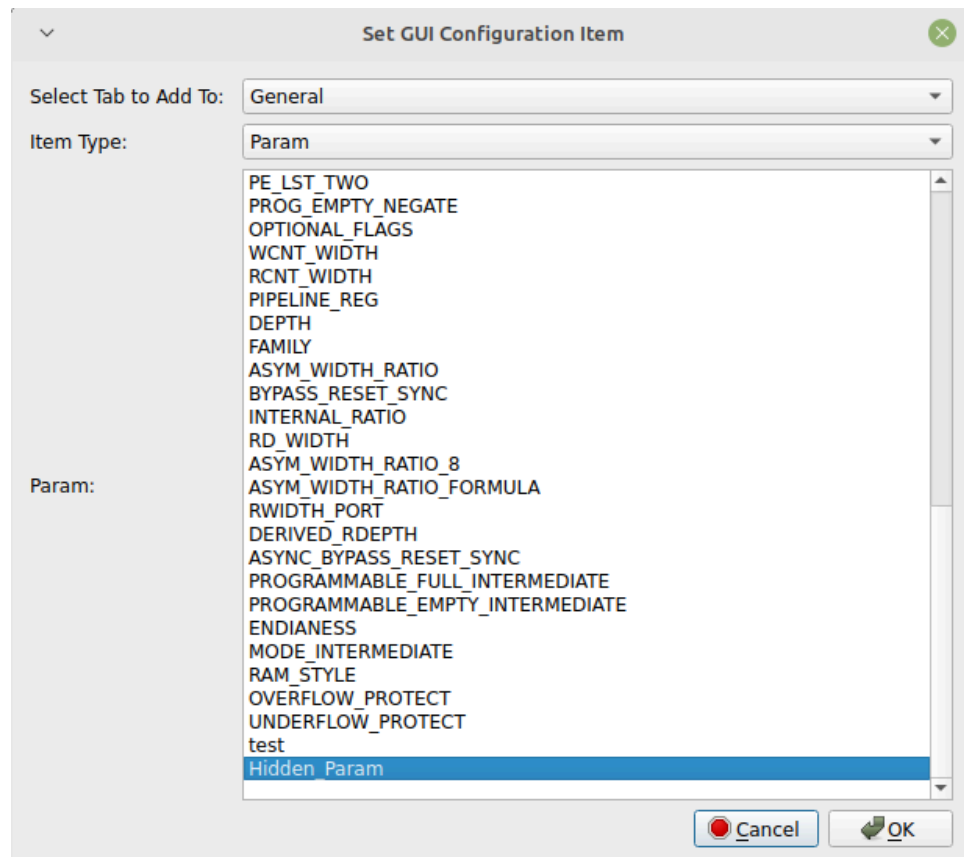
Figure 31: Drag and Drop Elements



You can drag and drop **NOTE_1** to after **DATA_WIDTH** to show more information about the **DATA_WIDTH** parameter.

When you add a parameter with the **Resolve** option as **immediate**, the IP Packager does **NOT** add it to the GUI Configuration tab automatically. You can add these parameters manually. Click the small green plus icon, choose **Item Type > Param**. The **Param** box shows the list of parameters. Select the parameter from the list.

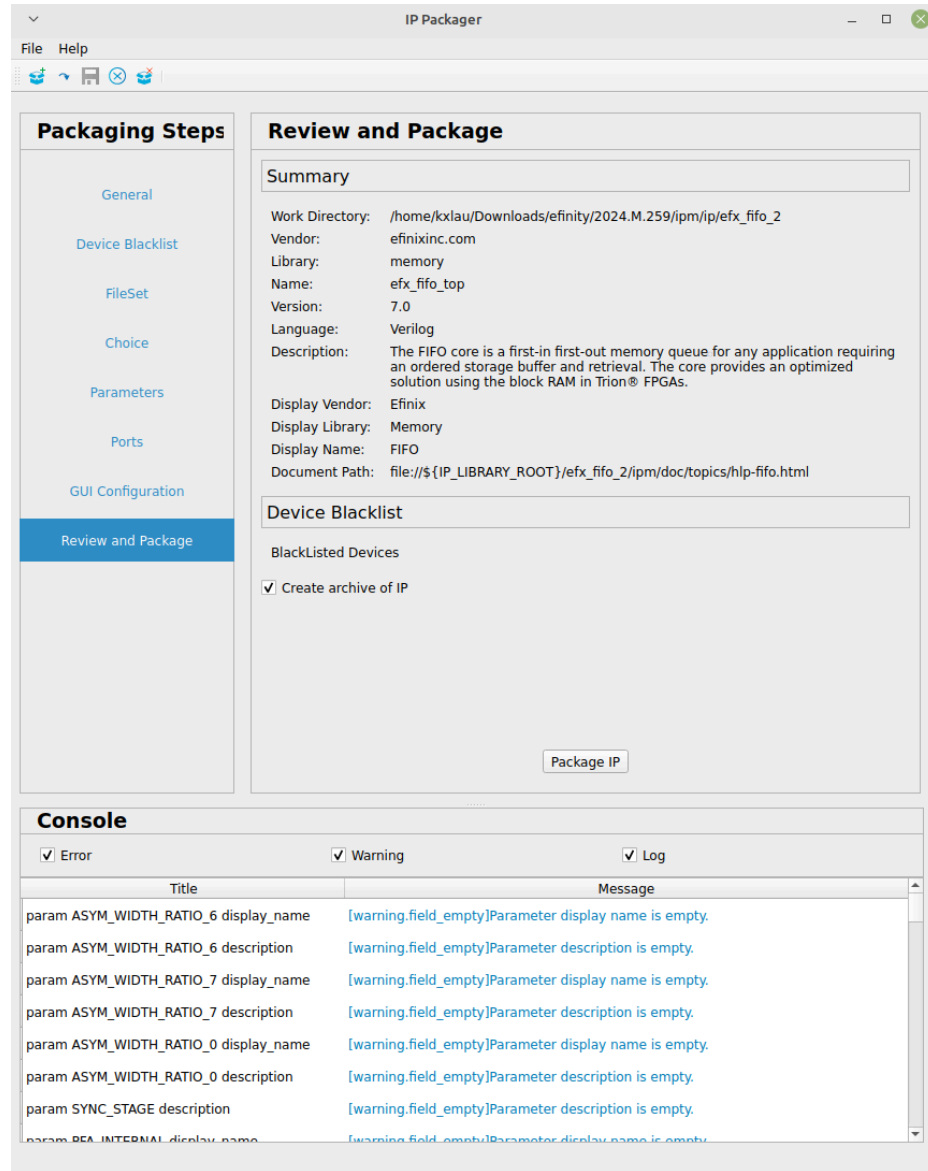
Figure 32: Add a Hidden Parameter



Review and Package Tab

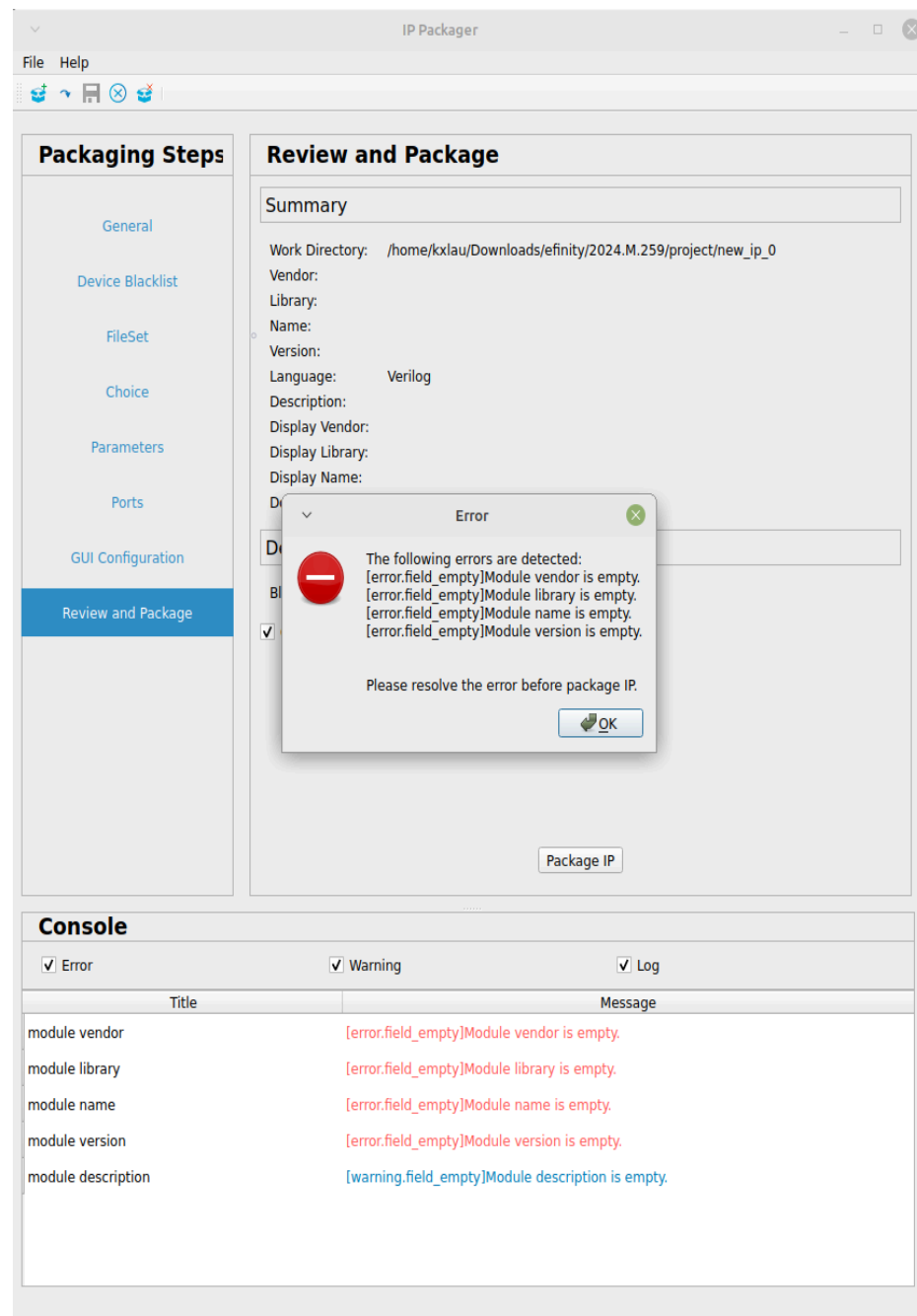
This tab allows you to check all your settings before packaging the IP. Once you have verified everything is correct, click **Package IP and Create Archive**. The IP Packager packages your IP and creates an archive (.zip) in your working directory.

Figure 33: Review and Package Tab



Clicking **Package IP** triggers a global check on all options. If there are any errors or exceptions in the console window, the IP Packager triggers a failure message and displays the failures in a pop-up window.

Figure 34: Example of Pop-Up Error Messages



Upon successful packaging, the `<working directory>/ipm` directory generates the following files and directories:

```
ipm (folder)
|---ip_component.xml
|---ip_component.xml.tmp
|---gui (folder)
| |---config_template.json
| |---config_template.json.tmp
```



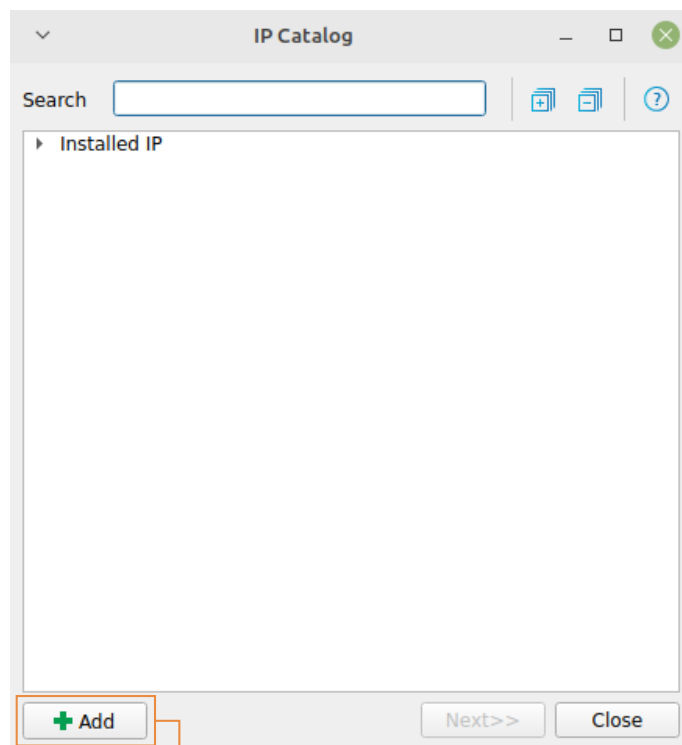
Note: The IP Packager packages all settings in the **GUI Configuration** page and saves them as **ipm/gui/config_template.json**. It packages other settings into the **ip_component.xml** file.

Import Self Packaged IP into IP Manager

To import self packaged IP into IP Manager, follow these steps:

1. Click on the **Add** tab in the **IP Catalog** and include the archived zip file. You can see your IP listed under the installed IP treeview in respective order of **Display Vendor Name**, **Display Library Name**, and **Display Name**.
2. Click on the **Display Name** > **Next** to open the IP Configuration page for this IP.

Figure 35: Import Self Packaged IP



Click on 'Add' to include the archived zip file

Expressions

IP Packager has a powerful expression handler that can recognize a variety of inputs.

Type Literal

A type is recognized based on the input types (`int`, `real`, `string`, `bit`, `unsigned`, and `signed`). The types correspond to the valid Verilog HDL types.

```
int      -> -2,147,483,647 to 2,147,483,647
string   -> anything start with ", end with ", eg "abc"
bit      -> eg 1'b0, 1'b1
unsigned -> eg 8'b10101010, 8'h56, 8'd170
signed   -> eg 8'sb10101010, 8'sh56, 8'sd170, 8'sd255
```

- `bit` is a single bit of either `1'b0` or `1'b1`.
- `unsigned` is a valid representations of Verilog HDL unsigned type.
- `signed` is a valid representation of Verilog HDL signed type.

Both unsigned and signed types represent binary, hexadecimal, and decimals:

Table 7: Data Types

Base	Signed	Unsigned
binary	8'sb10101010	8'b10101010
decimal	8'sh56	8'h56
hexadecimal	8'sd170	8'd170

Arithmetic Operator

The expression handler recognizes these arithmetic operations are recognized:

```
addition      -> a + b
subtraction   -> a - b
multiplication -> a * b
division      -> a / b
power         -> a ** b
```

These operations are interchangeable when handling `int`, `bit`, `unsigned`, and `signed` types. However, many calculations are resolved as a signed decimal, particularly for arithmetic operations between different data types, such as `int` and `unsigned`. For example:

```
8'sb11111111+99 = 8'sd255 + 99 = 33'sd98.
```

The signed decimal presents negative numbers in a non-negative way and standardizes the output in any data type or operation.



Note: Efinix does not encourage arithmetic operations between different data types; these operations produce warnings to the console even though they can be resolved.

Relational Operator

The expression handler recognizes these relation operations:

```
equal          -> a == b
not equal       -> a != b
larger than     -> a > b
smaller than    -> a < b
larger or equal -> a >= b
smaller or equal -> a <= b
```

Logical Operator

The expression handler recognizes these logical operators:

```
and      -> a && b
or       -> a || b
negation -> !a
```

Reference Operator

You can reference certain values from the **existing ip_component.xml**.

```
parameter reference ->    ${param}
global var reference ->   @ {global var}
current selection of global vars: @ {EFXDEVICE}, @ {EFXFAMILY}
```

Referencing Already Created Parameters

You can reference already created parameters. If the parameters do not exist, the **Console** shows an error. If the parameters exist, an evaluation is carried out.

```
Already created param: WEIGHT = 8'd20
Right now I wish to reference it in the new param,
ROUGH_MASS that I'm creating.
ROUGH_MASS.value = ${WEIGHT}*8'd10
                  = WEIGHT.value * 8'd10
                  = 8'd20 * 8'd10
```

Your value is based on other parameters' values. However, the parameters must be from the same **ip_component.xml** and must be created already.

Referencing Global Variables

When referencing global variables, you are explicitly referencing global variables that should only be set on start up. Reference global variables with the @ symbol.

```
The only current global variables that are available are:
EFXDEVICE = Project device, i.e. T8F81
EFXFAMILY = Project device family, i.e. Trion

Example:
IsPresent = @ {EFXDEVICE}=="T8F81"&&@ {EFXFAMILY}=="Trion"
```

Conditional Operator

The expression handler can understand conditionals.

```
condition ? expr_if_true : expr_if_false
```

Conditionals are helpful when you need to reference a parameter or some other global variable.

```
CURR_PARAM.value = ${OTHER_PARAM}<5 ? 32 : 64
```

Supported Function Calls

The expression handler recognizes these function calls:

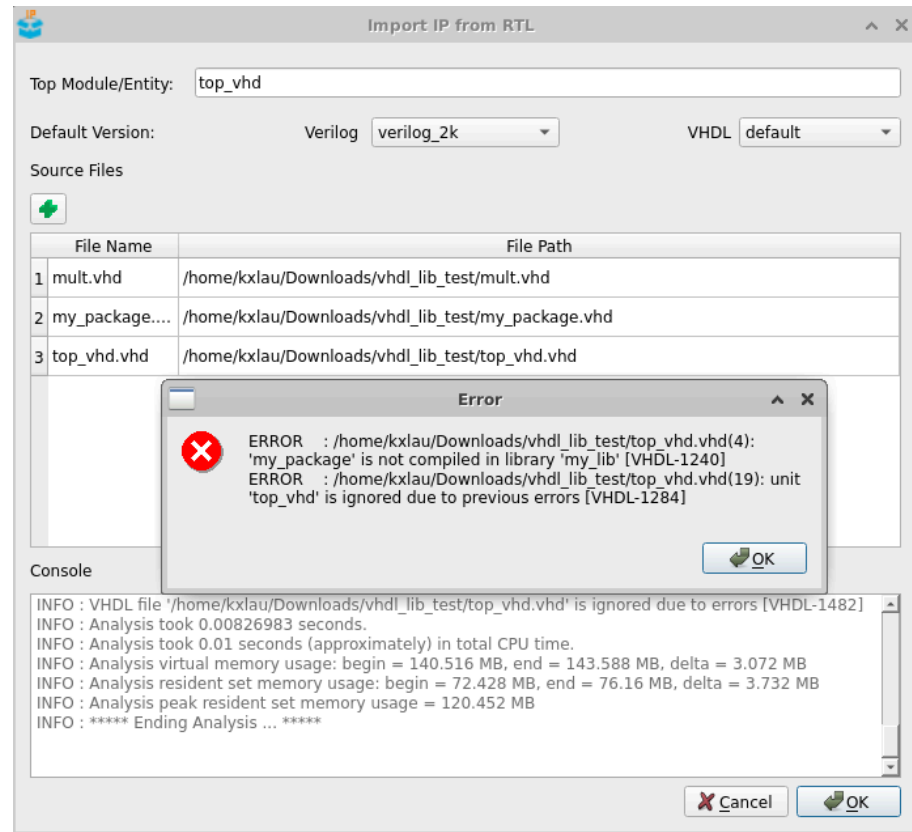
```
abs()           -> Get absolute value of a number, resolved to integer
min([])        -> Get the min number, given a list of int/bit/unsigned/signed. No spaces
                allowed inside the list.
max([])        -> Get the max number, given a list of int/bit/unsigned/signed. No spaces
                allowed inside the list.
log2()         -> Get the log2 of a number, resolved to integer. Decimal values are
                floored.
int()          -> Convert a given number into an integer. Strings are currently invalid
                input.
str()          -> Convert an input into a string.
upper()        -> Convert a given string into an uppercase string
lower()        -> Convert a given string into a lowercase string
```

```
Examples: abs(-5) = 5
abs(1'b1)         = 1
abs(8'b1101)     = 13
min([1, 2])      = Invalid syntax error, due to space after comma
min([2,5,4])     = 2
min([6,4'd4])    = 4'd4
max([1, 2])      = Invalid syntax error, due to space after comma
max([2,5,4])     = 5
max([8'b1101,4]) = 8'd13
log2(1'b1)       = 0
log2(16)         = 4
log2(13)         = 3
log2(8'b1101)   = 3
int(8'b1101)     = 13
int(5.60)        = 5 # NOTE: Real values are currently not selectable in IP Packager
int("3")         = Error. String cannot be converted to int currently.
str(-5)          = "-5"
str(1'b1)        = "1'b1"
str(8'sb111)     = "8'sd7"
upper(-4)        = Not valid as integer is not a string
upper("hello")   = "HELLO"
lower(7)         = Not valid as integer is not a string
lower("HELLO")  = "hello"
```

Using Custom VHDL Libraries

The IP Packager does not support custom VHDL libraries when you import RTL files. The tool shows an error if your VHDL RTL code depends on code from a custom VHDL library.

Figure 36: VHDL Custom Library Error



In this example, the tool cannot find **my_package** in **my_lib** because **top_vhd.vhd** defines **my_package** as being in the library **my_lib**. Additionally, the information is stored in a project **.xml**. The IP Packager cannot create a custom library. The following code shows an example.

```
# top_vhd.vhd
library ieee;
use ieee.std_logic_1164.all;

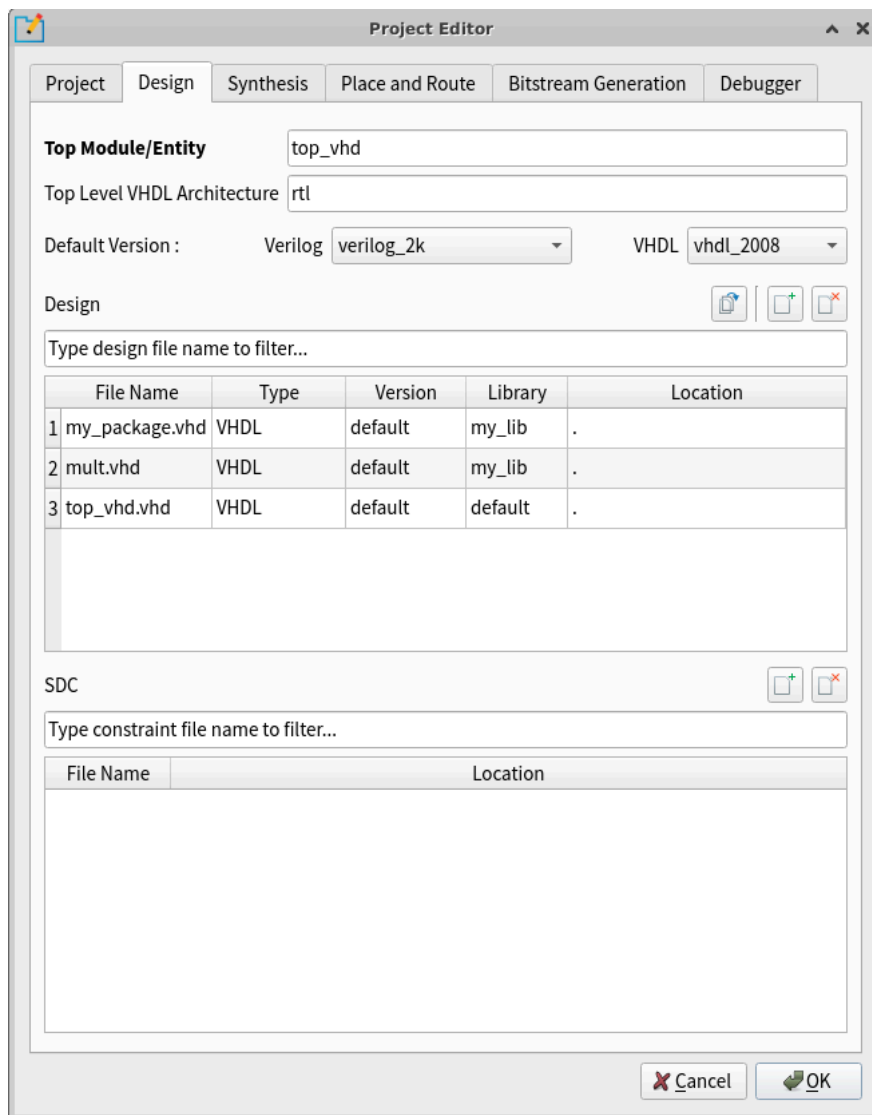
library my_lib;
use my_lib.my_package.all;
```

However, you can still use custom VHDL libraries in the Efinity software by following these steps:

1. Create an Efinity project.
2. Include your design files and relevant project details.
3. Click **OK**.

4. If your RTL has a defined custom library, add it and the related files to your project. For example, the **my_package.vhd**, **mult.vhd**, and **top_vhd.vhd** where **my_package.vhd** and **mult.vhd** are part of the custom library **my_lib**.

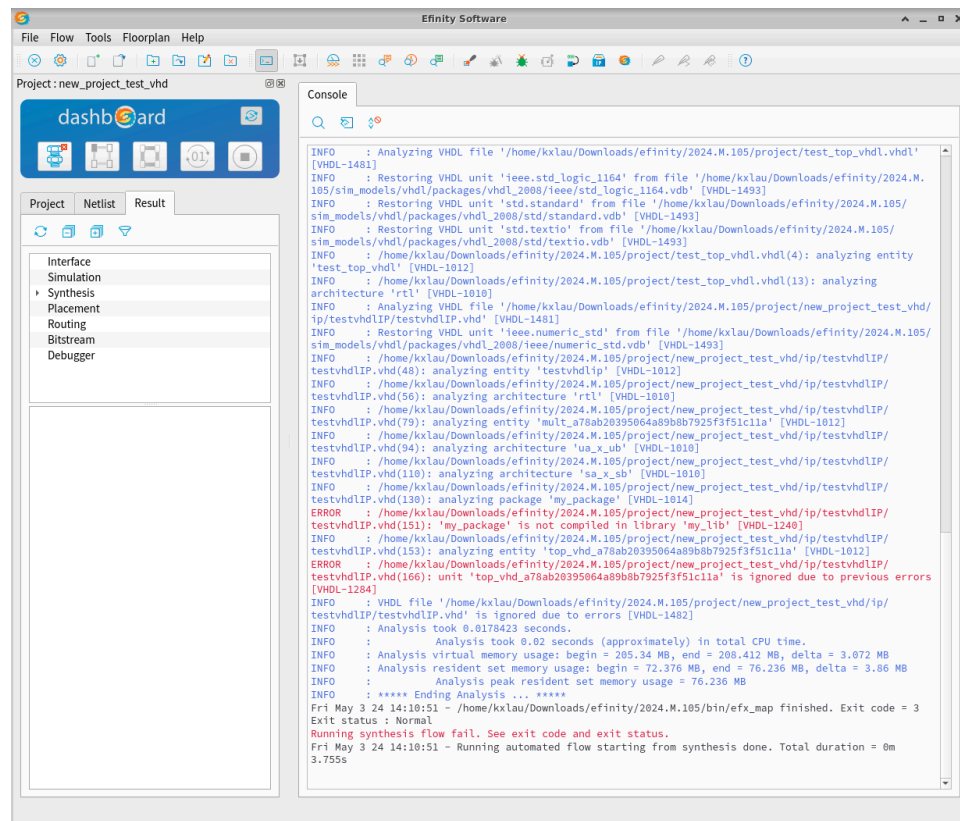
Figure 37: Efinity Project Editor with Custom Library



5. Click **OK** and refer to the following figure.
6. Right-click **Project tab > Design** and choose **Package** to open the IP Packager.
7. The IP Packager and Import IP page opens with all the design files and top module name included. If you click **OK**, you will not encounter errors in the RTL custom library.

However, when you create a new project, import the packaged VHDL IP into the IP catalog, and generate the IP, you might face the custom library error again when you instantiate the VHDL IP in this new project because the **efx_map** could not find the library information.

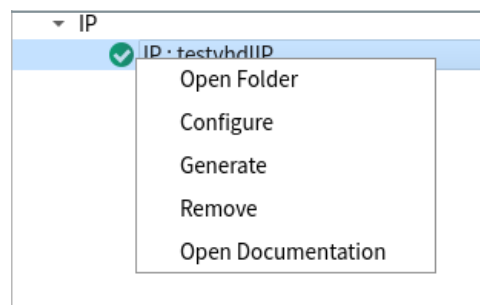
Figure 38: Error in Instantiating an IP with Custom Library



The **efx_map** could not find the library information because there is no library information in this new project XML. The reason is that there is still custom library information inside the generated IP top wrapper. The RTL codes are concatenated into a generated top wrapper file during the IP generation. A temporary solution is provided.

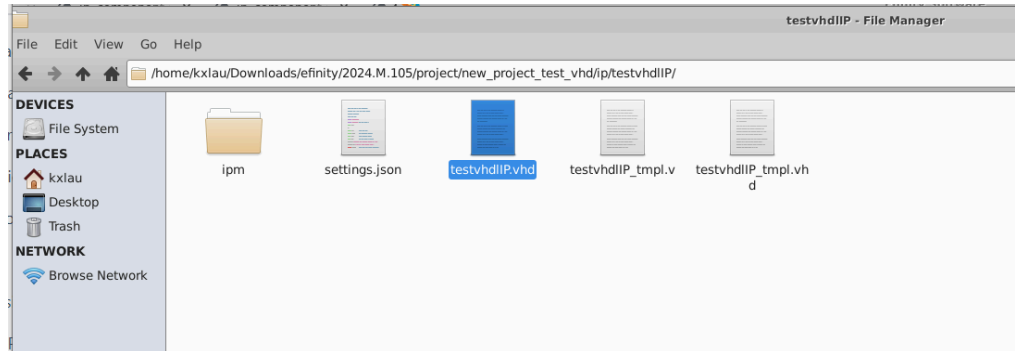
1. You can right click on the IP and choose **Open Folder**.

Figure 39: Opening a Folder



- Click on the top wrapper file, **ip_name.vhd** to open the file in your preferred text editor (e.g., **testvhdIIP.vhd**).

Figure 40: Example opening a Top Wrapper File



- You will see the following code:

```
library my_lib;
use my_lib.my_package.all;
begin
  inst_mult: entity
  my_lib.mult_a78ab20395064a89b8b79225f3f51c11a(sa_x_sb)
  generic map
  (
    AW => AW,
    BW => BW
  )
end;
```

- Manually replace all the library names with **work**'. For example, the custom library name is **my_lib**, and you can change all references of **my_lib** to **work** using the text replace feature of your text editor in the top wrapper file.

```
library work;
use work.my_package.all;
begin
  inst_mult: entity work.mult_a78ab20395064a89b8b79225f3f51c11a(sa_x_sb)
  generic map
  (
    AW => AW,
    BW => BW
  )
end;
```

- Save the file and go back to Efinity to compile your project again. The compilation shows success if the design is valid.



Note: Using 'work' as the library name is just one kind of identifier referring to the current library.

Revision History

Table 8: Revision History

Date	Version	Description
November 2024	1.2	<p>Added information in Fileset Tab topic. (DOC-2151 & DOC-2010)</p> <p>Removed the part on the introduction on ways of using the IP Packager in the Open the IP Packager topic and also information from General Tab topic.</p> <p>Changed title from Importing New IP Using the RTL Scanner to Packaging New IP Using the RTL Scanner. Added a statement in the topic. In this topic, the sub-topic is Import an Existing Configuration and Save.</p> <p>Updated image in Parameters Tab, Custom Rule Checker, Ports Tab, and Review and Package Tab.</p>
July 2024	1.1	<p>Updated Importing New IP Using the RTL Scanner. (DOC-1990)</p> <p>Changed the title from Entering Data from Scratch to Setting Up the IP.</p> <p>Moved Save topic after Import an Existing Configuration.</p>
June 2024	1.0	Initial release (DOC-1826).