Creating a custom IP in Vivado

I. Create the module

Open Vivado 2016.4 and create a new project. Create a new VHDL file called logic_function.vhd.

We will first create a 1-bit Logical AND. In the "*Define Module*" define 2 1-bit inputs (A_inp and B_inp) and 1 1-bit output.

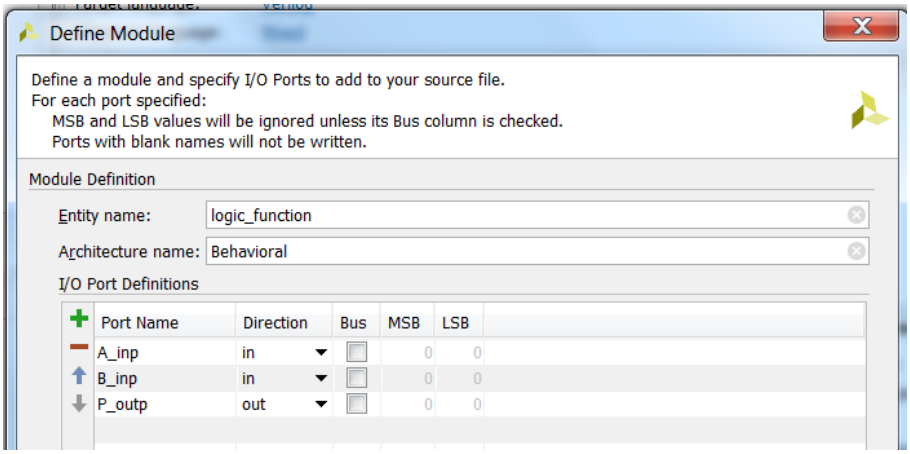

Figure 1 - Define the module

In the architecture of the module, just add the following line:

```
P_outp <= A_inp AND B_inp;
```

II. Package the module in an IP

Click on "*Tools > Create and Package New IP…*" to create a new project to package an IP. In the "Create Peripherals, Package IP or Package a Block Design" page, select "*Package your current project*".

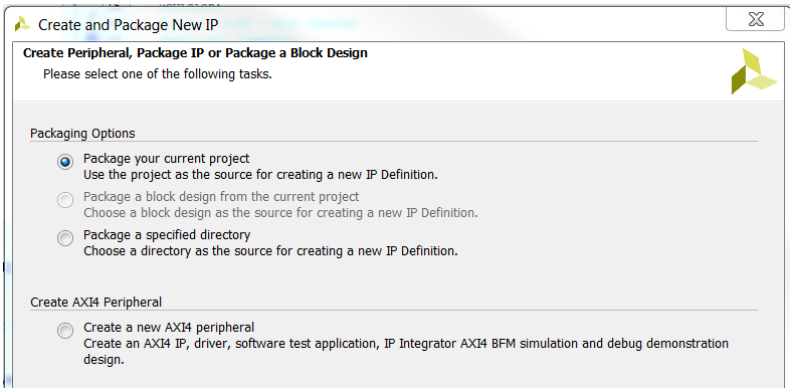

Figure 2 - Package your current project

In the "*Package your current project*" page, choose where you want the IP to be created and select "*Include .xci files*". This create a temporary project where you can edit the IP configuration. You

should have a window called "*Package IP – logic_function*" opened with various sections. The section "Identification" for example allow you to change how the IP will be identified in the Vivado IP catalog.

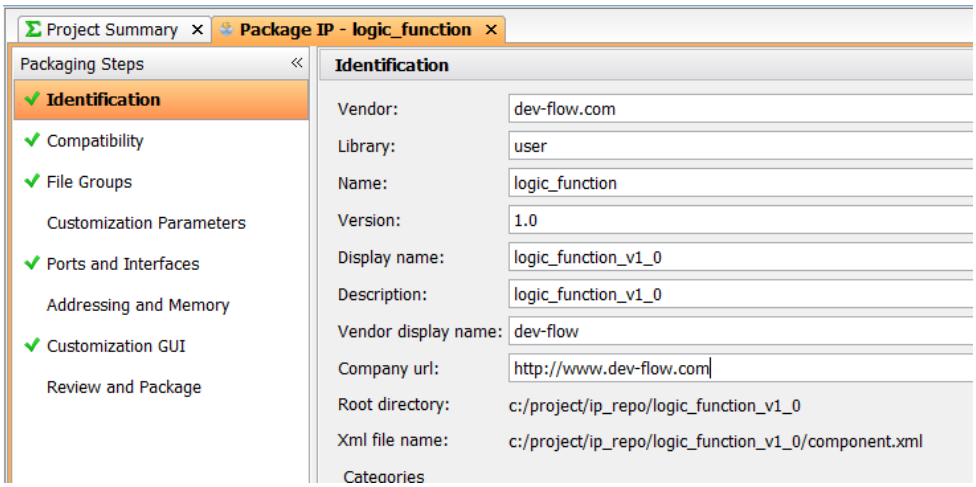

Figure 3 - Package IP – Identification

We won't configure the IP for the moment. Go in the section "*Review and Package*" and click on "*Package IP*". Vivado will ask you if you want to close the project, click "*Yes*".

Then, in your initial project, if you go in the IP catalog ("*Window > IP catalog*") you should see the IP under UserIP.

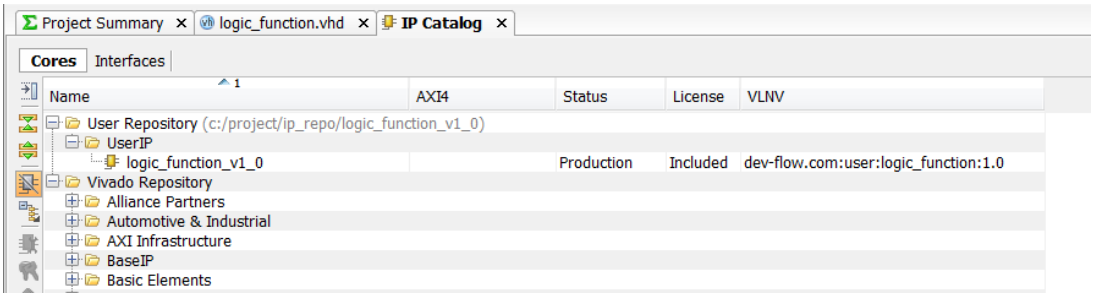

Figure 4 - User IP

> *Note: If you create a new Vivado project, you won't see the IP in the user catalog. This is because the folder where the IP is located is not scanned by Vivado. You need to add this folder to the "IP repositories" in the Project Settings (IP section > Repository Manager Page).*

III. Modifying the IP

Now, we will modify the IP to do a logical AND on buses (not only on 1-bit signals). To modify the IP, open the IP catalog and search for the IP. Right click on the IP and click on "Edit in IP Packager".

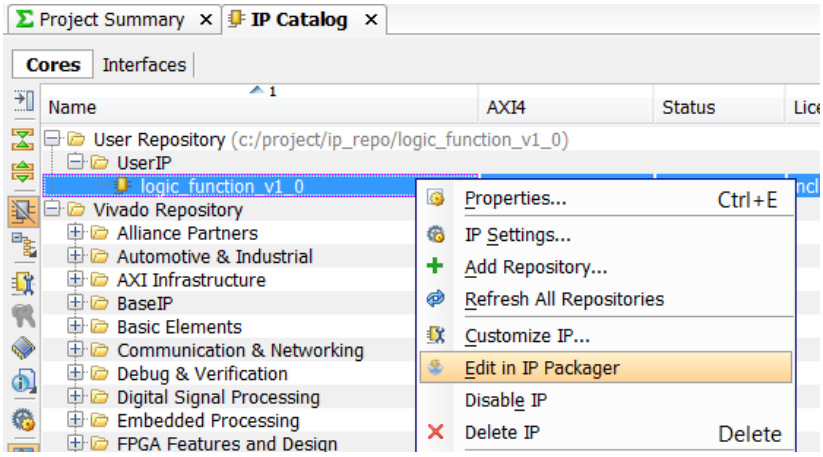

Figure 5 - Edit in IP Packager

In the design sources, doubled click on the source file (logic_function.vhd) to modify it.

Change the entity adding generics values to allow different size of inputs.

```
entity logic_function
   Generic (
      DATA_SIZE : INTEGER := 32
   );
   Port (
      A_inp : in STD_LOGIC_VECTOR(DATA_SIZE-1 downto 0);
      B_inp : in STD_LOGIC_VECTOR(DATA_SIZE-1 downto 0);
      P_outp : out STD_LOGIC_VECTOR(DATA_SIZE-1 downto 0)
   );
end logic_function;
```

Go to the "*Package IP*" window, in the section "*Customization Parameters*". Click on "*Merge Changes from Customization Parameters wizard*" to have the changes taken in account in the IP.

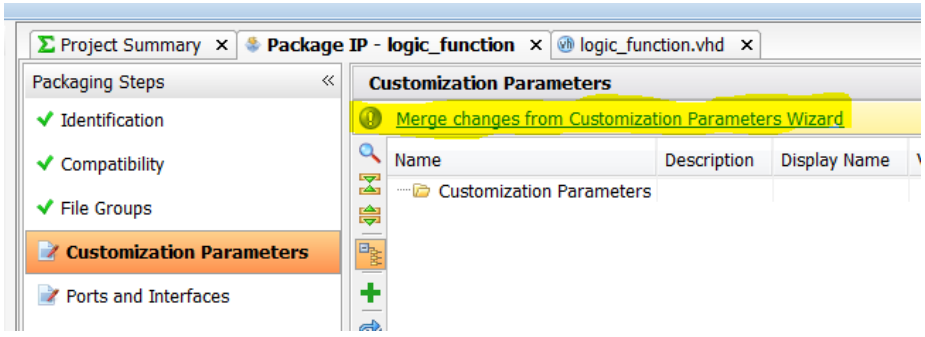

Figure 6 - Merge Changes from Customization Parameters wizard

Then, under "*Hidden Parameters*", double click on "*DATA_SIZE*". In the window "*Edit IP Parameter*", enable "*Visible in customization GUI*" to be able to change the value when adding the IP to a project from the IP catalog.

Then enable "*Specify Range*" and select "*Range of Integer*" for type. Enter "*32*" as maximum value and click OK.

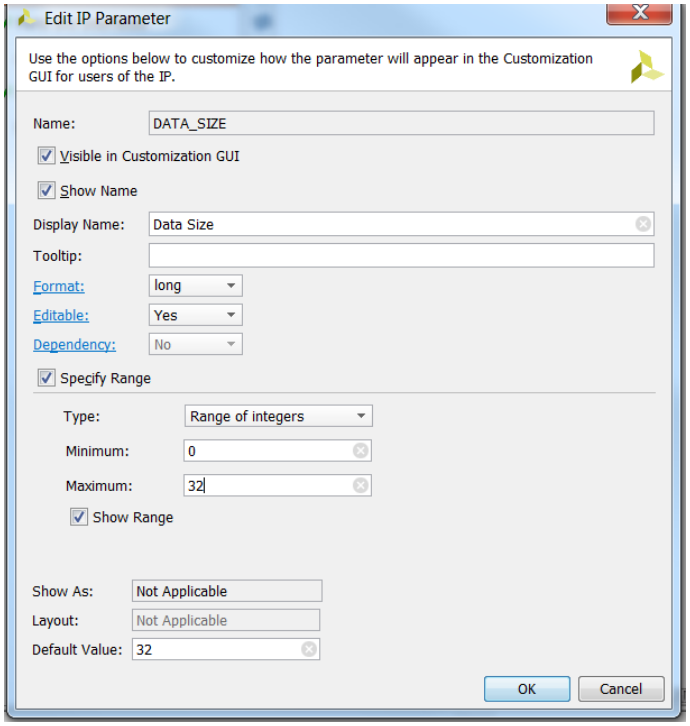

Figure 7 - Edit IP Parameter

Then re-package the IP in the "*Review and Package*" section.

Now, if we add double click on the IP in the IP catalog, we can see that we can select the size of the data in the GUI.

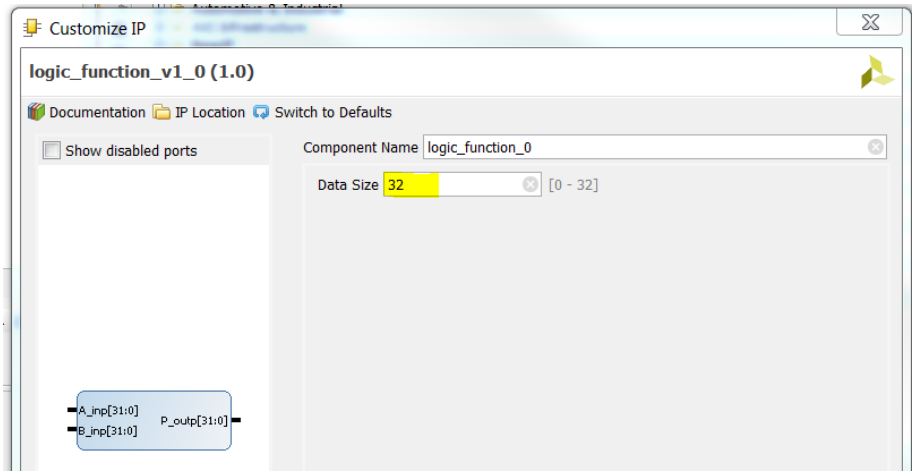

Figure 8 - IP GUI

IV. Adding a different functionality

Now, we will add other functions to our IP: logical OR, logical NOR and logical NAND. And the user will be able to select the functionality he wants to use when adding the IP to a project.

Open the IP in "*IP Packager*" and open the source code and modify it with the following code.

```
entity logic_function is
   Generic (
      DATA_SIZE : INTEGER := 32;
      FUNC_SEL : INTEGER := 0
   );
   Port (
      A_inp : in STD_LOGIC_VECTOR(DATA_SIZE-1 downto 0);
      B_inp : in STD_LOGIC_VECTOR(DATA_SIZE-1 downto 0);
      P_outp : out STD_LOGIC_VECTOR(DATA_SIZE-1 downto 0)
   );
end logic_function;

architecture Behavioral of logic_function is

begin

   IF_AND: IF(FUNC_SEL = 0) GENERATE
      P_outp <= A_inp AND B_inp;
   END GENERATE;

   IF_OR: IF(FUNC_SEL = 1) GENERATE
      P_outp <= A_inp OR B_inp;
   END GENERATE;

   IF_NOR: IF(FUNC_SEL = 2) GENERATE
      P_outp <= NOT(A_inp OR B_inp);
   END GENERATE;

   IF_NAND: IF(FUNC_SEL = 3) GENERATE
      P_outp <= NOT(A_inp OR B_inp);
   END GENERATE;

end Behavioral;
```

Save the source file and merge the changes in the "*Package IP*" window. Edit the parameter FUNC_SEL:

- enable "Visible in customization GUI"
- Specify range:
    - Pairs
    - Min: 0
    - Key: And, Value: 0
    - Key: Or, Value: 1
    - Key: Nor, Value: 2

- Key: Nand, Value: 3

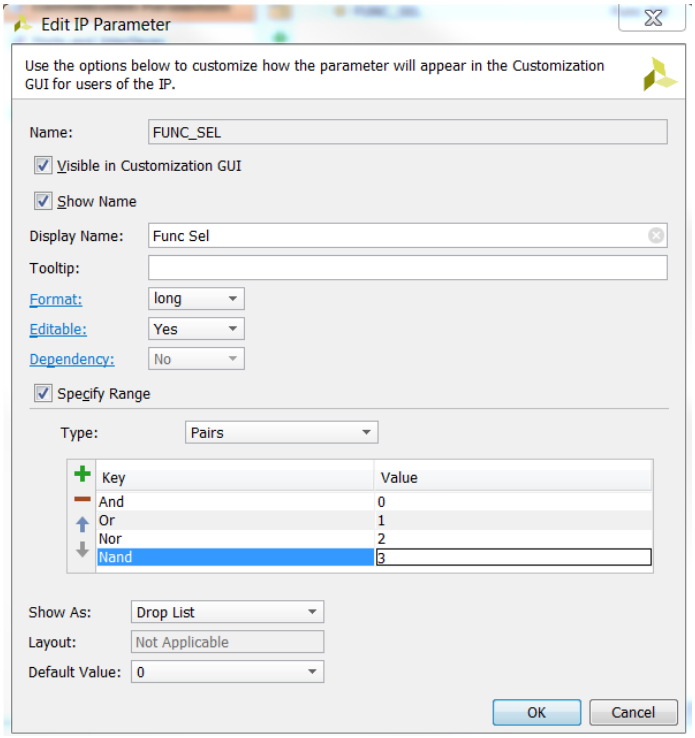

Figure 9 - Customization of parameter FUNC_SEL

Validate and re-package the IP.

Now, from the IP GUI, you can select which function you want.

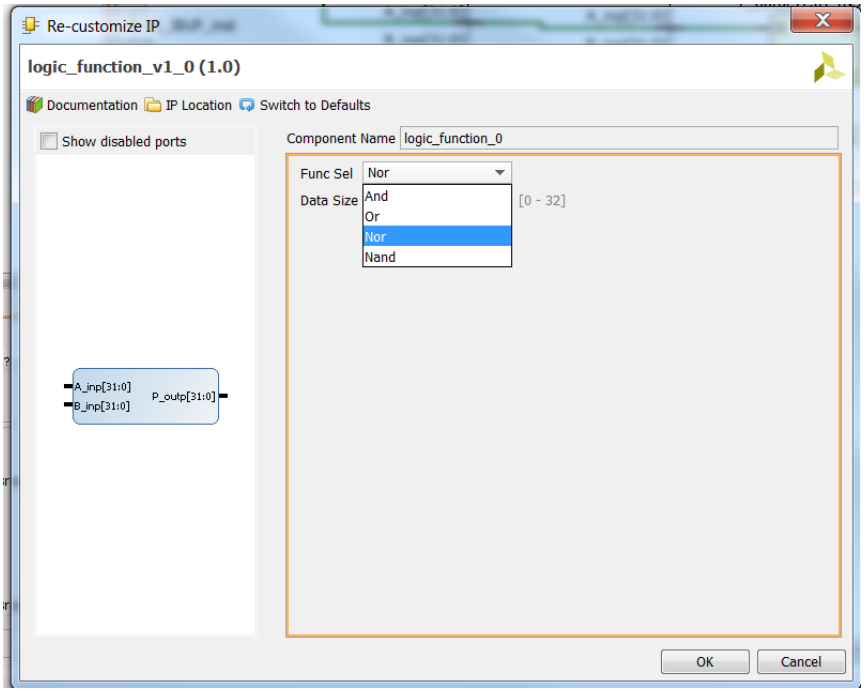

Figure 10 - Function selection in the IP GUI

V. Unused port

We will now add a new function to our IP: a logic NOT. For this we only need one input port. The other one will be unused so we will disable it from the IP.

Open the IP in "*IP Packager*" and open the source code and modify by adding the following code.

```
IF_NOT: IF(FUNC_SEL = 4) GENERATE
   P_outp <= NOT(A_inp);
END GENERATE;
```

Save the source file and merge the changes in the "*Package IP*" window. Edit the parameter FUNC_SEL by adding a new pair (key: Not, Value: 4).

In the "Ports and Interfaces" section, double click on B_inp to open the "*Edit Port*" window. If the user want to use the logical NOT function, the input port B_inp can be useless. So in change to "*Optional*" for "*Port Presence*" and type "$FUNC_SEL != 4" in the box below as shown in the Figure 11.

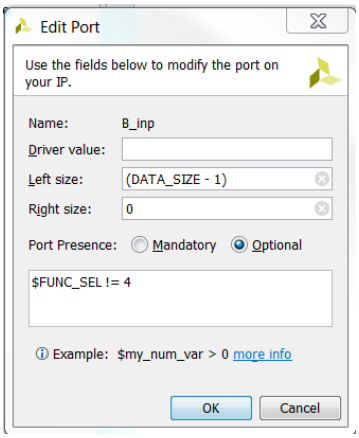

Figure 11 - Edit port B_inp

Vivado will display the following warning:

[IP_Flow 19-3161] Port 'B_inp': The tie-offs driver value is not specified for the input port with an enablement condition.

This is warning is just to tell that we haven't defined a value for when the port B_inp is disabled. This warning could be ignored because when the port B_inp is disabled, it is also not used in the IP. But if you want to remove this warning, you can add a default value to the port in the component declaration:

```
B_inp : in STD_LOGIC_VECTOR(DATA_SIZE-1 downto 0) := (others => '0');
```

Merge the changes in the "*Package IP*" window again if needed and then re-package the IP.

Now in our original project, we can see that if we select the function “Not”, the input port B_inp will be disabled for the IP.

