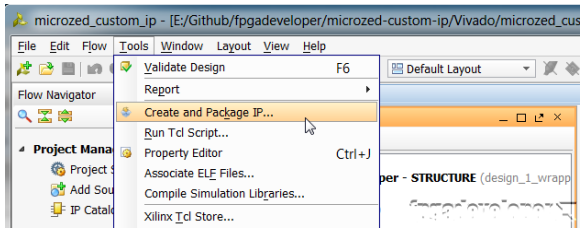
Create a custom IP Block with AXI Interface

http://www.fpgadeveloper.com/2014/08/creating-a-custom-ip-block-in-vivado.html

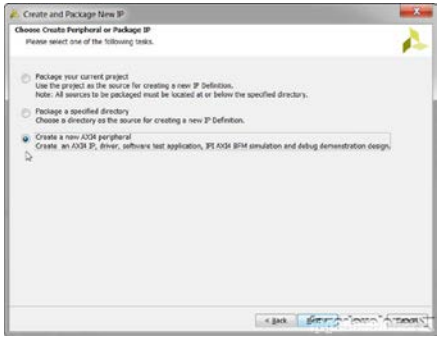
Dr. Heinz Rongen
Forschungszentrum Jülich GmbH
Zentralinstitut Systeme der Elektronik (ZEA-2)
H.Rongen@fz-juelich.de

Create IP Block

- Create a basic SOC Design
  - Create Block diagram / Add IP / ZYNQ7 PS
  - Run Block Automation

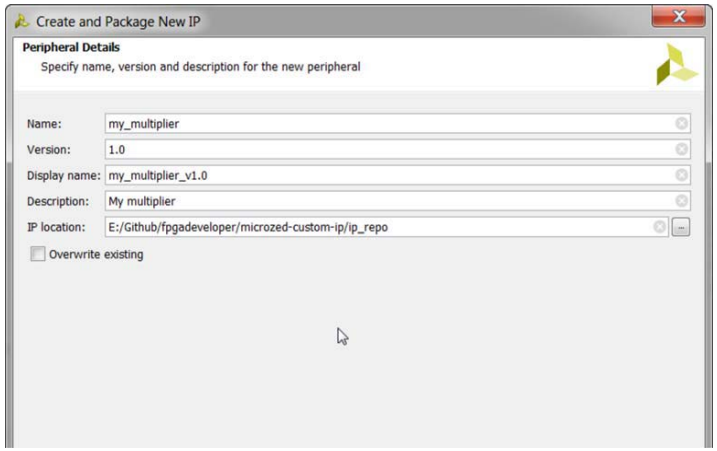


- Now: **Tools → Create and package IP**
- Click Next



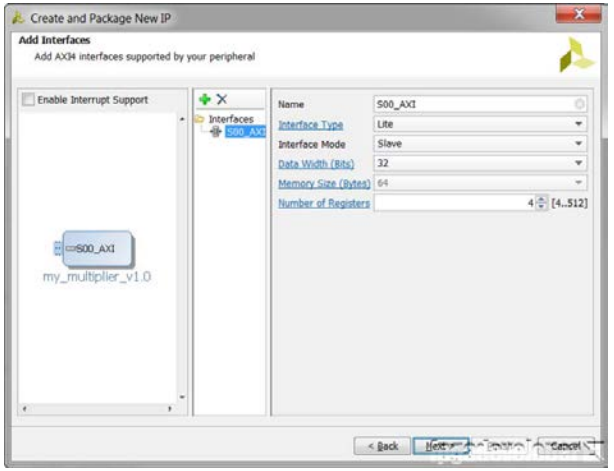
- Click “Create a new AXI4 peripheral”

- Give a name to your new IP Block, Description and location on the disc

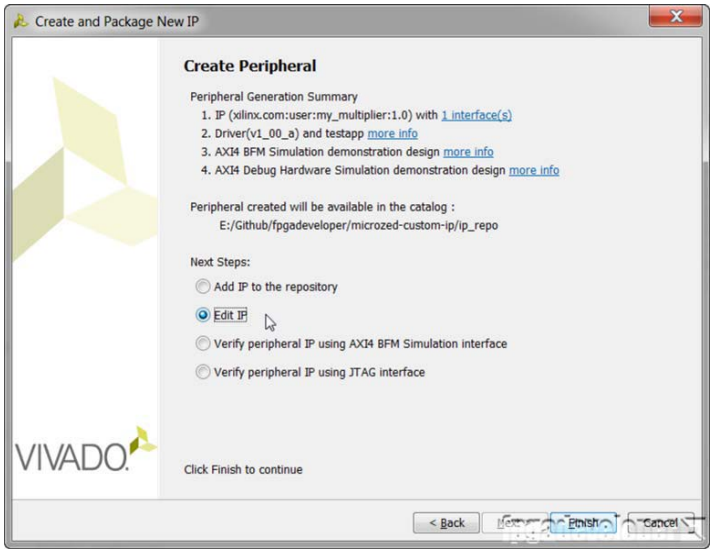


Configure IP Block / AXI interface

- Configure the IP Block, the AXI bus interface
  - AXI Lite, a Slave, Bus width 32 bit (defaults are ok for this example)



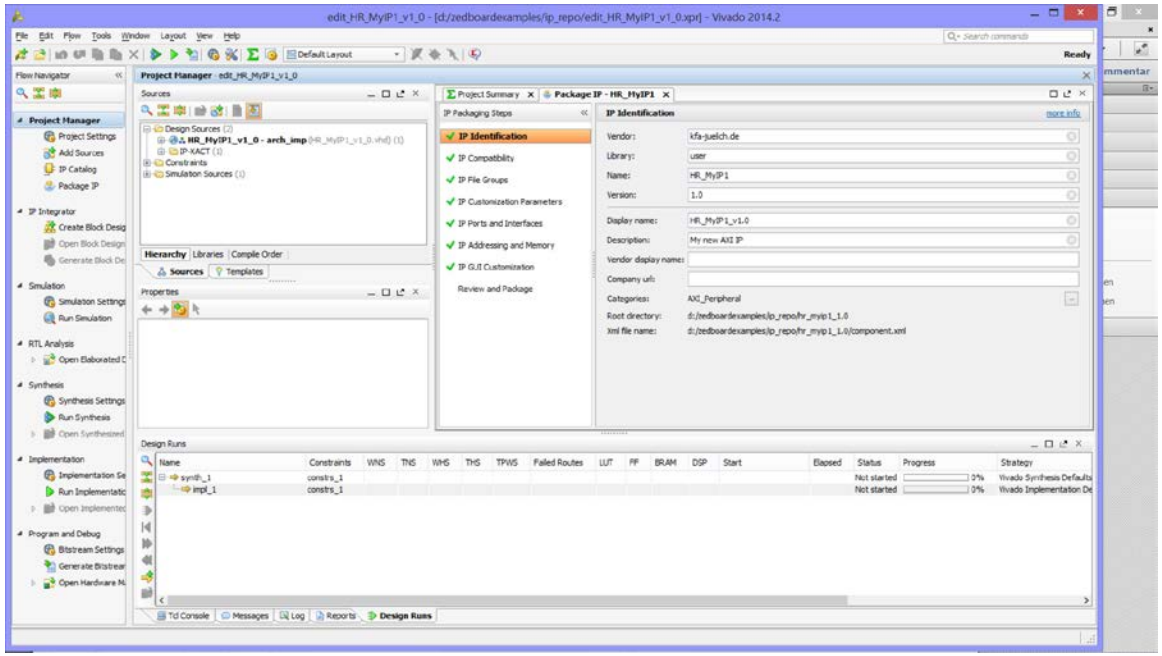
- The next page is a summary
  - Select “Edit IP”
  - Click Finish



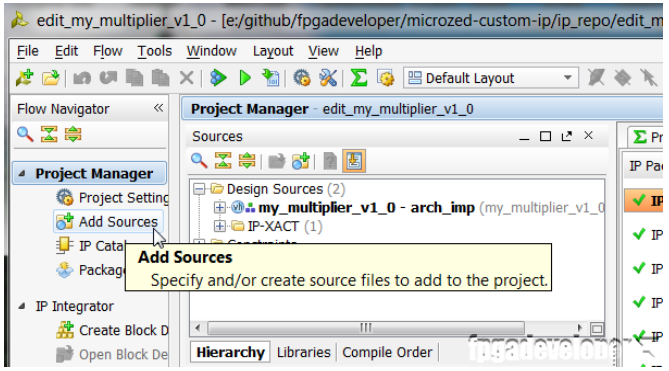
Another Vivado window will now open

Open the VHDL source

A second Vivado instance is open, to modify (describe) our IP Block



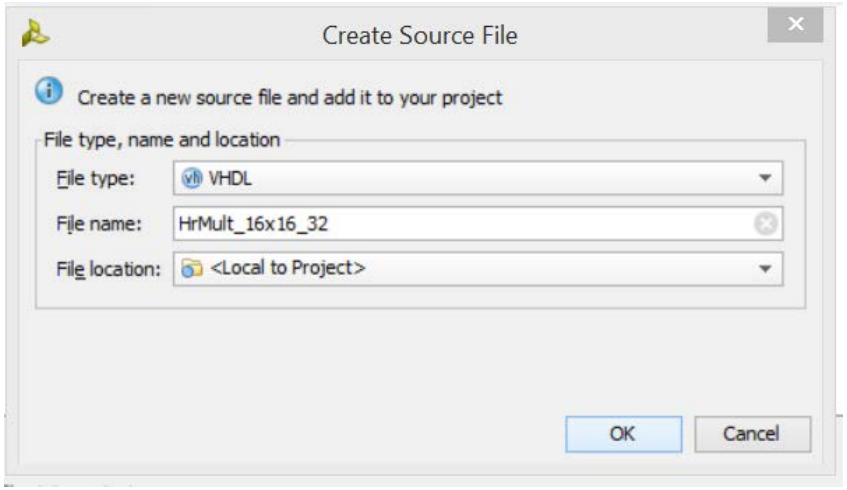
- In the Flow Navigator
- Click “Add Source”



- Click on “Add or Create Design Sources”
- Next

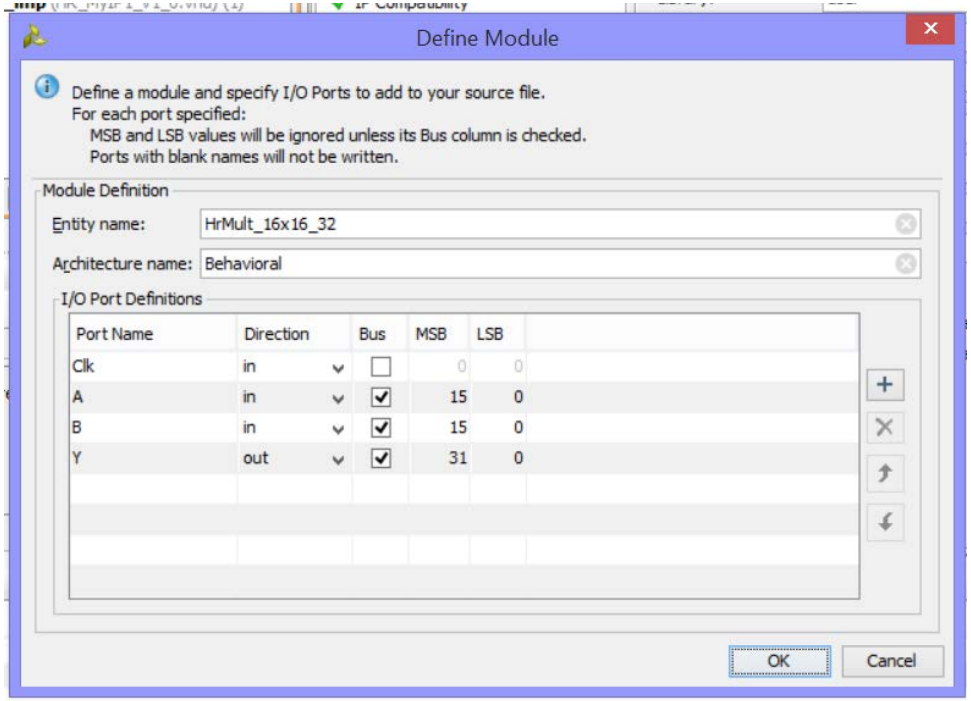
Define Entity

- Select VHDL
- Type a name for the VHDL source
- Click OK
- Click “Finish” (in the page before)



VHDL module creation wizard

- Define the entity
- as traditional VHDL

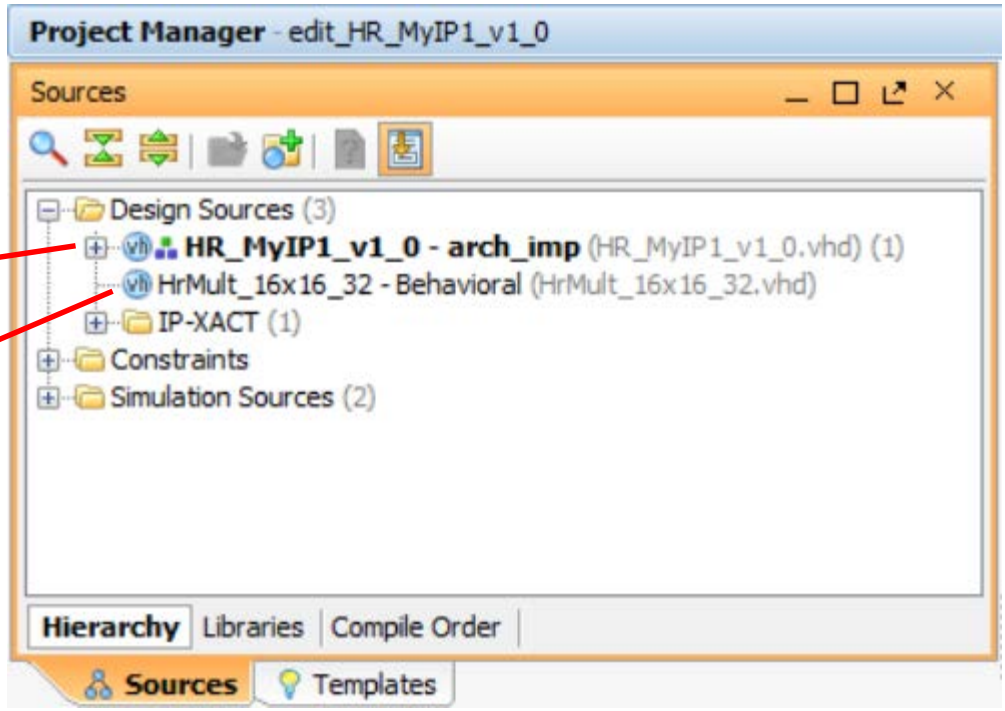


-

- The Project Manager should now looks like this

The IP Core (AXI Interface)

The VHDL implementation



Instantiate the VHDL code (our behavior) inside the IP Core

- Double click on the “name**_AXI_inst**”

Declaration

- Find the line with the “begin” keyword
- add the following code just **above** it (in the VHDL declaration part)
- and declare the multiplier component

```
signal multiplier_out : std_logic_vector(31 downto 0);

component multiplier
port (
  clk: in std_logic;
  a: in std_logic_VECTOR(15 downto 0);
  b: in std_logic_VECTOR(15 downto 0);
  p: out std_logic_VECTOR(31 downto 0));
end component;
```

Instantiation

- Now find the line that says "– **Add user logic here**" and add the following code below it to instantiate the multiplier:

```
multiplier_0 : multiplier
port map (
  clk => S_AXI_ACLK,
  a => slv_reg0(31 downto 16),
  b => slv_reg0(15 downto 0),
  p => multiplier_out);
```

- Find this line of code "reg_data_out <= slv_reg1;"
- and replace it with "reg_data_out <= multiplier_out;".

- In the process statement just a few lines above, replace "slv_reg1" with "multiplier_out".

- Save the file.

You should notice that the "multiplier.vhd" file has been integrated into the hierarchy because we have instantiated it from within the peripheral.

VHDL source code for multiplier

Find VHDL code

https://github.com/fpgadeveloper/microzed-custom-ip/blob/master/Vivado/ip_repo/my_multiplier_1.0/src/multiplier.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity multiplier is
  port(
    clk : in std_logic;
    a   : in std_logic_vector(15 downto 0);
    b   : in std_logic_vector(15 downto 0);
    p   : out std_logic_vector(31 downto 0)
  );
end multiplier;

architecture IMP of multiplier is

begin
  process (clk)
  begin
    if clk'event and clk = '1' then
      p <= a * b;
    end if;
  end process;
end IMP;
```