System-On-FPGA Partial Reconfiguration Tutorial

March 2015

Introduction

This tutorial shows how to develop a partial reconfiguration design from the bottom up using the Xilinx® Integrated Synthesis Environment (ISE), Xilinx® Platform Studio (XPS), Software Development Kit (SDK), and the PlanAhead design tool. ISE will be used to create the low-level Reconfigurable Modules (RM). In this tutorial, one of the RM will be a binary count up and the other RM will count down. All this will be seen on the LEDs. The XPS will be used to create a processor hardware system that includes a lower-level module defining one Reconfigurable Partition (RP) and the two Reconfigurable Modules (RM). The SDK will be used to create a software application that enables you to perform partial reconfiguration. This tutorial uses the Processor Configuration Access Port (PCAP) Interface to perform the partial reconfiguration that is stored in the RAM and SD card. The PlanAhead help with the Floorplan of the design.

For this tutorial the Zedboard Evaluation Kit is used to implement the partial reconfiguration. The Zedboard is a complete development kit with an integrated Xilinx Zynq®-7000 All Programmable SoC. The ISE Suite 14.7 was used to build the design.

At the end of this tutorial the Zedboard will have two partial reconfigurable bit files stored in the SD card. At boot-up the Zedboard should transfer the bit files to the DDR Ram. In software, PCAP will read those bit files and reconfigure the hardware. The hardware will show the interactions, by illuminating the LEDs. The LEDs will act as a binary counter, starting by incrementing up. Whenever a button is pushed the hardware will be reconfigured and the LEDs will increment down.

Prerequisite

1. This tutorial assumes that the reader has basic experience with Xilinx Design Suite.

2. Folder Setup for simplicity.
   This step is to have a more organized folder layout to simplify some of the later steps. It will help avoid confusion.

   1. Create a project folder.

   2. Inside this project folder create 4 new folders with the name ise, edk, bitstream, and planahead.

   3. Inside the ise folder create 2 folders name bitcountup and bitcountdown

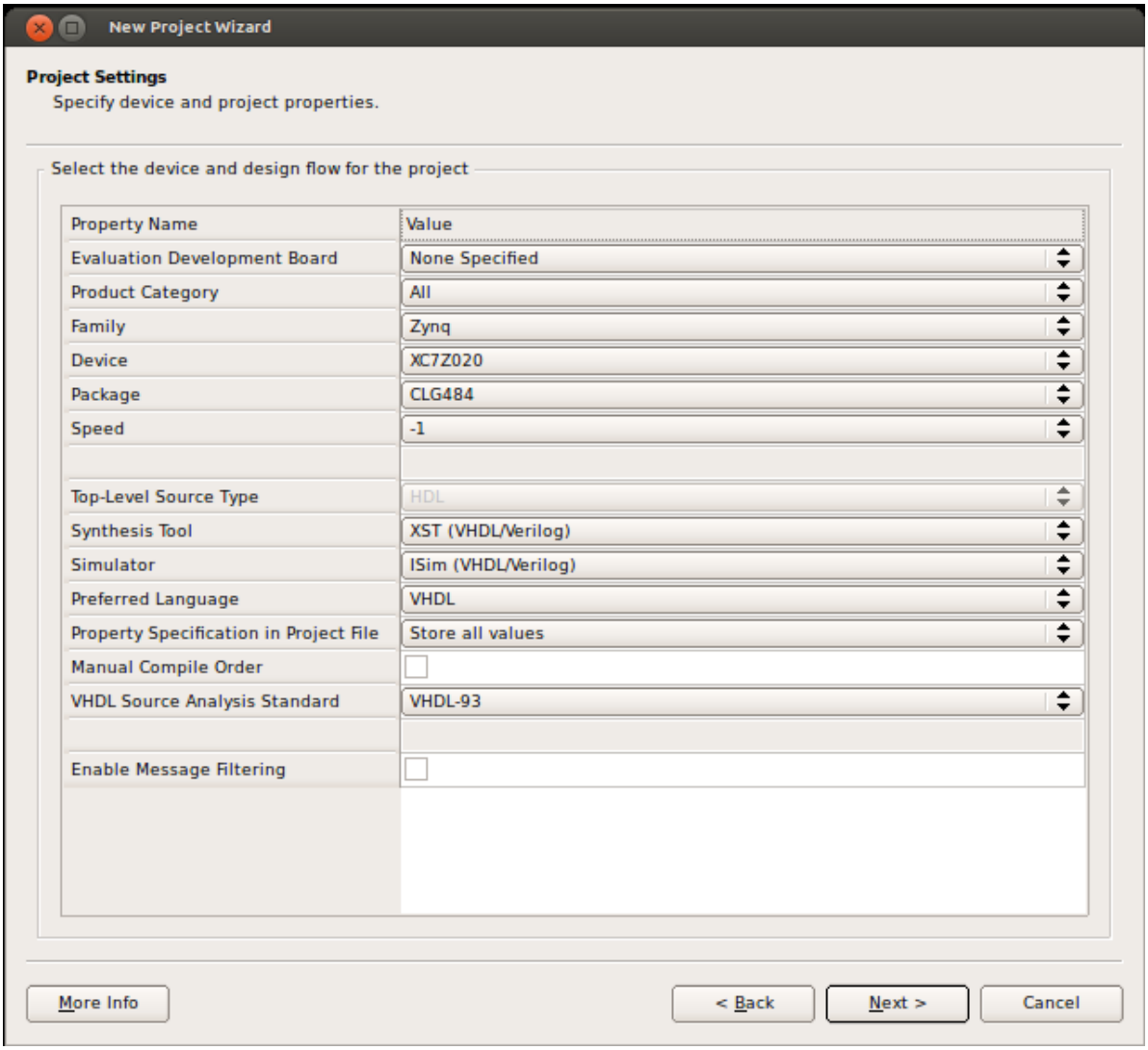
Creating the Partial Reconfiguration Modules

1. Open ISE and create a new project in the bitcountup folder.

2. Lets call this project prconfig

3. In the project wizard choose these settings.

Device Family: Zynq
Device: xc7z020
Package: clg484
Speed: -1

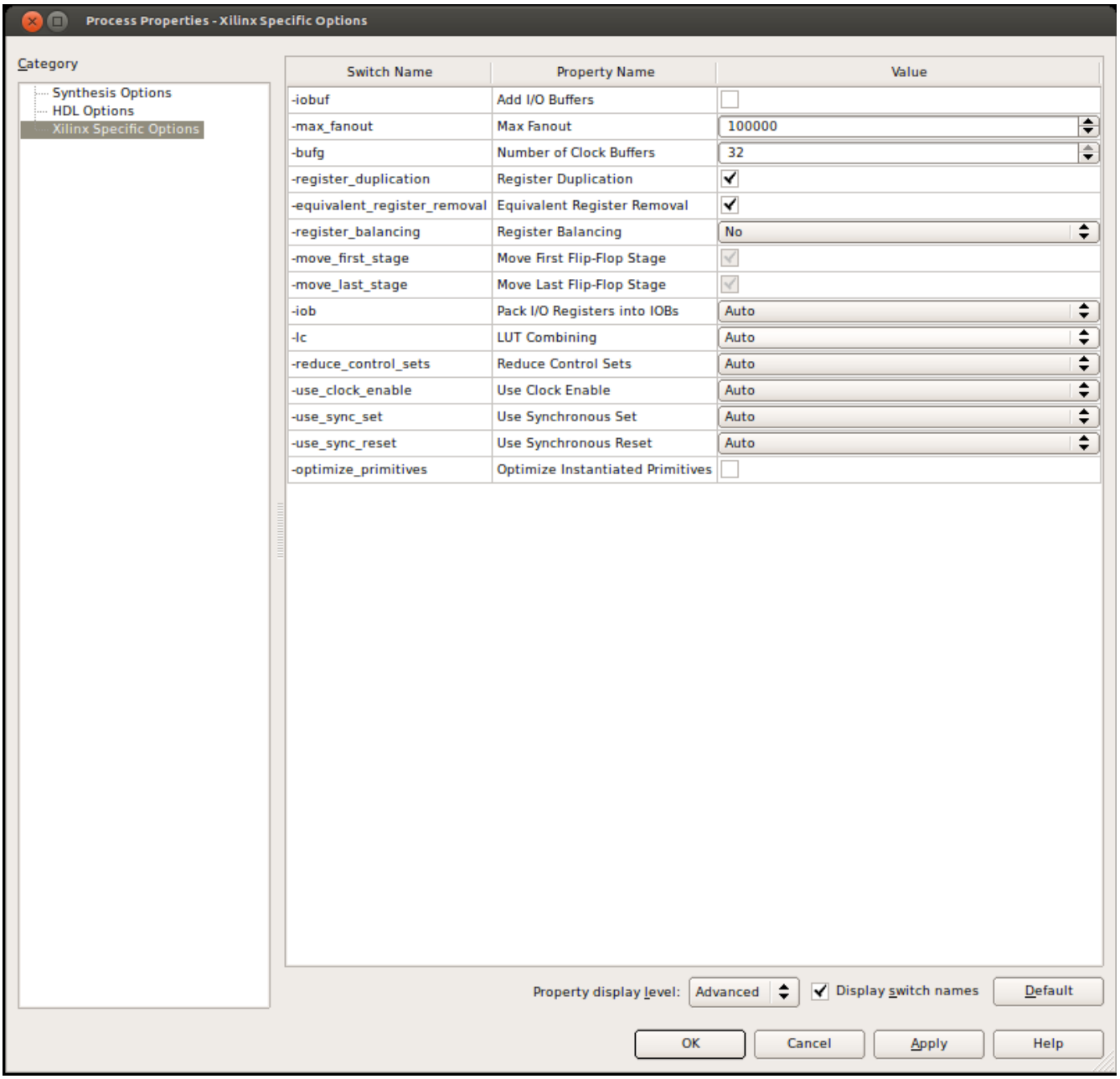


4. Add a VHDL module and name it prconfig
    1. Do not add any ports.
5. Copy this code to the the module

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;

entity prconfig is
    Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           input : in  STD_LOGIC_VECTOR (7 downto 0);
           output : out  STD_LOGIC_VECTOR (7 downto 0));
end prconfig;
architecture Behavioral of prconfig is
signal din,dout : STD_LOGIC_VECTOR(7 downto 0);
begin
           process(clk,reset)
           begin
                    if clk'event and clk = '1' then
                              din <= input;
                              if reset = '0' or din = x"FF" then
                                        din <= (others => '0');
                                        dout <= (others => '0');
                              else
                                        dout <= din + 1;
                              end if;
                    end if;
           end process;
output <= dout;
end Behavioral;
```

6. The module names for both module will have the same names.

7. They also will have the same input and the same output ports.



8. Click on the top module (prconfig) and in the Design Window right click Synthesize - XST then click Process Properties. When the Process Properties pops up click Xilinx Specific Options. Make sure that the IO buffer check box is unchecked.

9. Click okay and double click Synthesize - XST to create the .ngc files that will be implemented later.

Create the Second Partial Reconfiguration Module

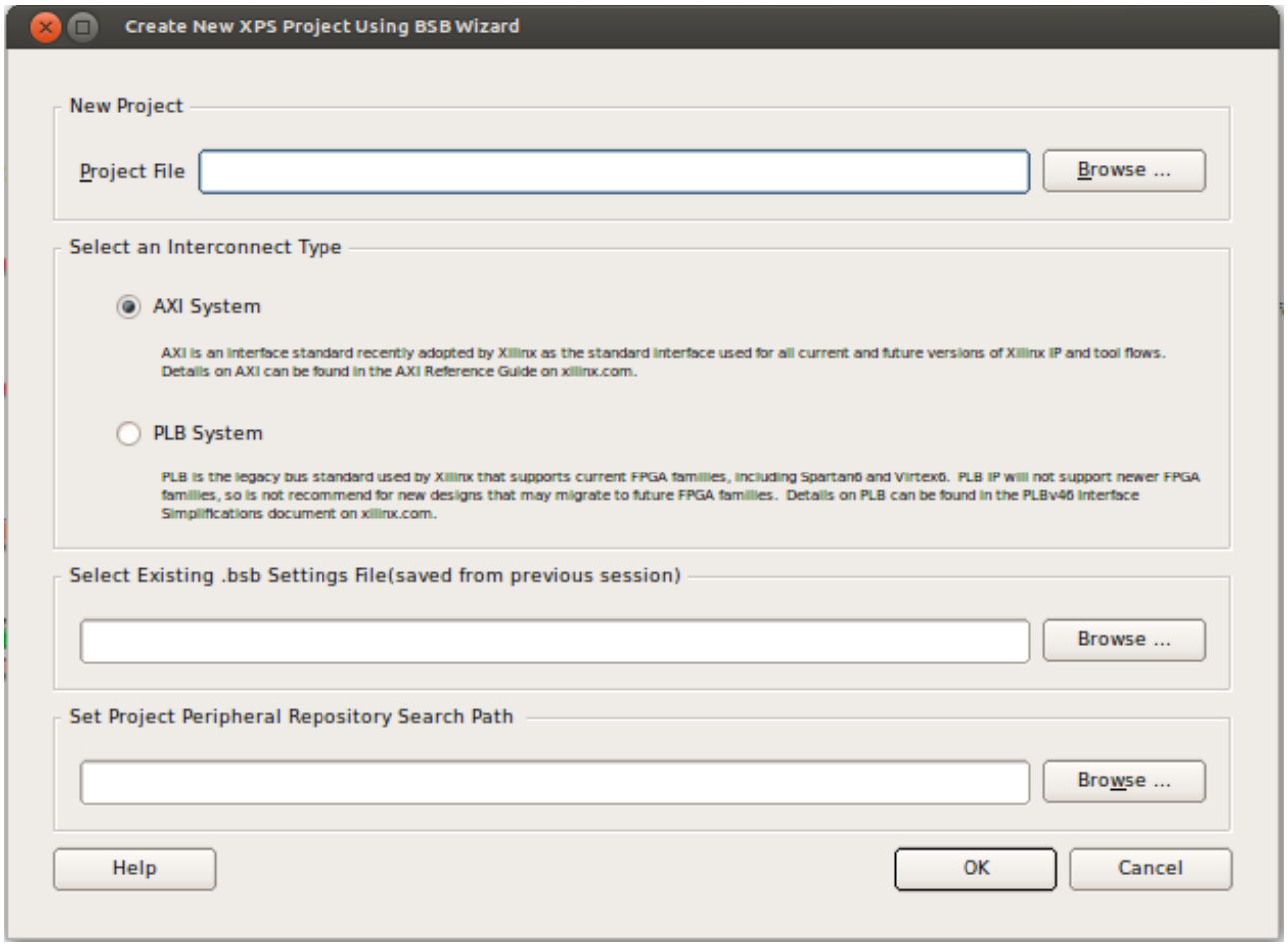
1. Repeat the First Partial Reconfiguration module steps except in the bitcountdown folder and your logic will be different but still consist of the same input and output.

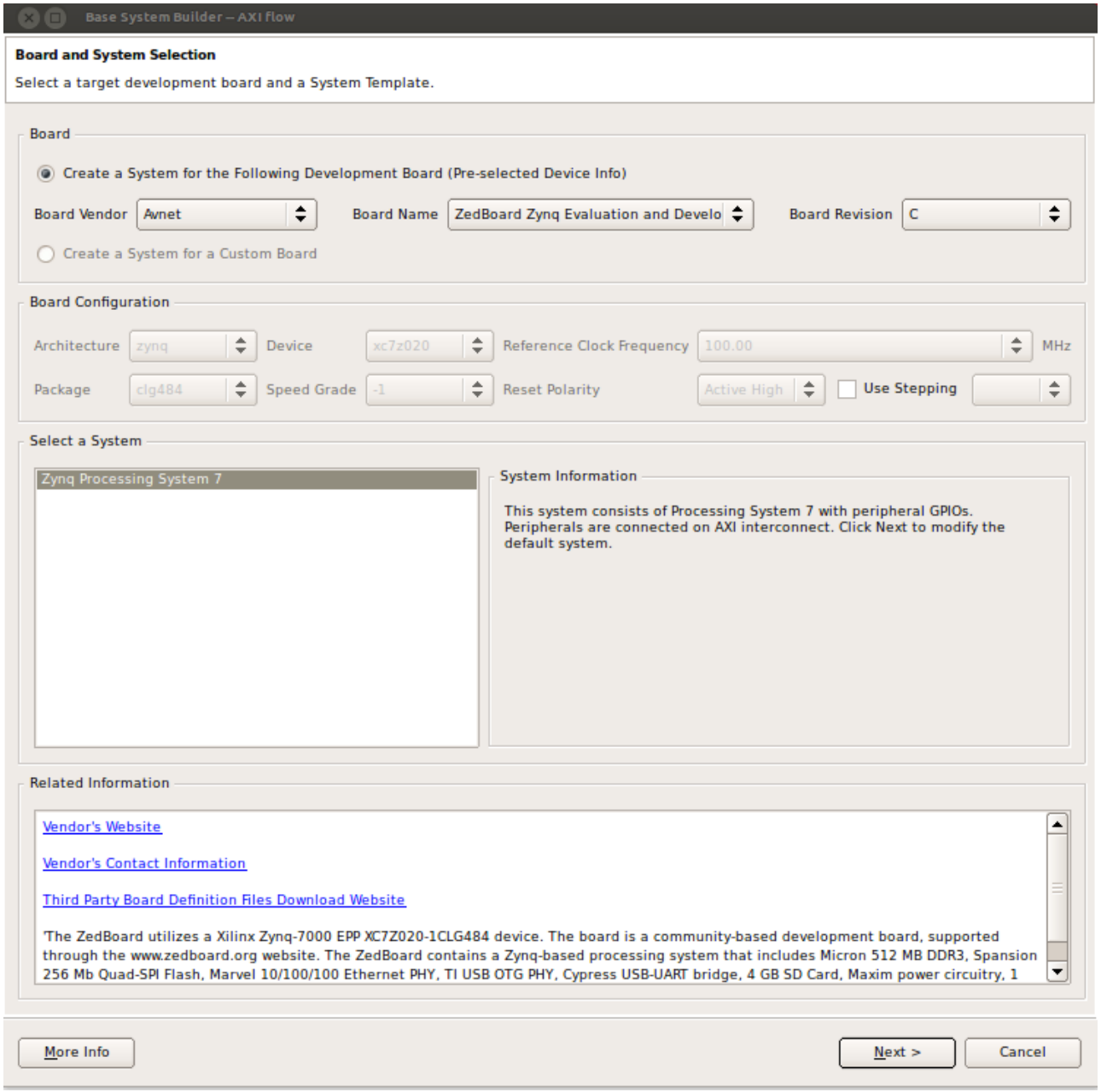
2. Copy this code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;

entity prconfig is
   Port ( clk : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          input : in  STD_LOGIC_VECTOR (7 downto 0);
          output : out  STD_LOGIC_VECTOR (7 downto 0));
end prconfig;

architecture Behavioral of prconfig is
signal din,dout : STD_LOGIC_VECTOR(7 downto 0);
begin

          process(clk,reset)
          begin
                    if clk'event and clk = '1' then
                              din <= input;
                              if reset = '0' or din = x"00" then
                                        din <= x"FF";
                                        dout <= x"FF";
                              else
                                        dout <= din - 1;
                              end if;
                    end if;
          end process;

output <= dout;
end Behavioral;
```

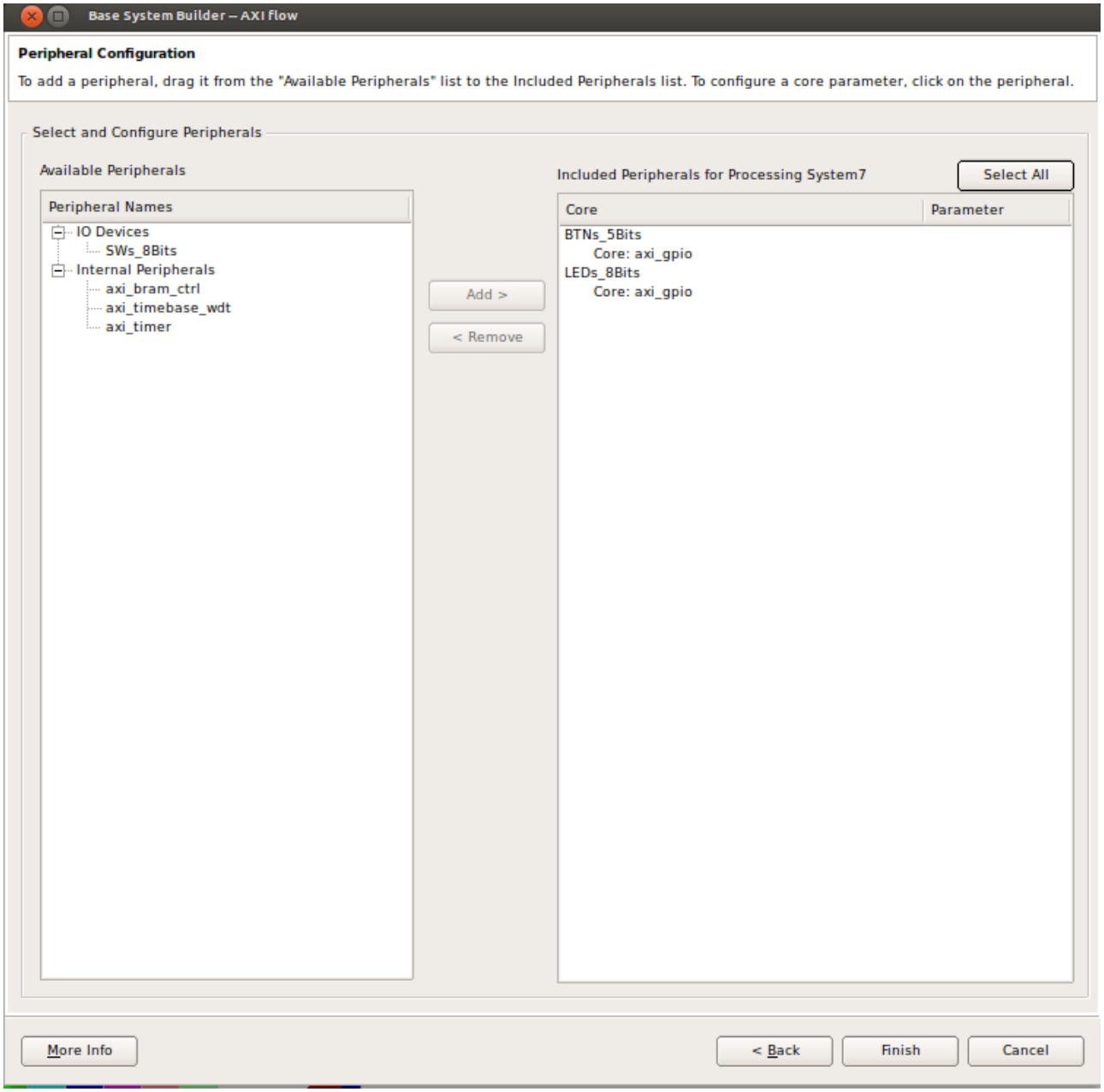
Create a Hardware/Software System in EDK

1. Create a new EDK project inside the edk folder using the Base System Builder. Save the system.xmp inside the edk folder. Keep the default settings for an AXI system.





2. When ask to select a target development board and a System Template select the Avnet Zedboard Zynq Evaluation and Development kit C.
3. In the next window when asked to add peripheral remove everything except the buttons and LEDs then click Finish.



Create a IP Peripheral Reconfigurable Partition

1. In the Toolbar click on Hardware and click Create and Import Peripheral Wizard. Click Next.
2. In the next screen make sure “Create a template for a new peripheral” is picked and click next.
3. Click next again using the default setting.
4. Name the peripheral bit_counter_ip then click next.
5. Keep the default settings to use the AXI4-Lite, click next 2 times.
6. Choose 2 registers then click next 2 more times to keep the default settings.

7. In the Peripheral Implementation Support window make sure that the “Generate ISE and XST project files to help you implement the peripheral using XST flow” box is checked then click finish.

Synthesizing the IP Peripheral in order to create a Black Box

1. Open the ISE project of the IP peripheral in edk/ipcore/bit_counter_ip/devl/projnav
2. In the Design Hierarchy double click on USER_LOGIC_I to open the design. Browse the code and under the “--USER signal declarations added here, as needed for user logic” add this code.

```
component prconfig port(clk : in STD_LOGIC;
     reset : in STD_LOGIC;
     input  : in STD_LOGIC_VECTOR(7  downto 0);
     output : out STD_LOGIC_VECTOR(7 downto 0));
end component;
```

3. Next search for “--USER logic implementation added here” add this next code:

```
prconfig_i : prconfig port map(Bus2IP_Clk,Bus2IP_Resetn,slv_reg0(7 downto 0),slv_reg1(7 downto
```

4. In the SLAVE_REG_WRITE_PROC process code replace the whole process with this code

```
  if Bus2IP_Clk'event and Bus2IP_Clk = '1' then
   if Bus2IP_Resetn = '0' then
    slv_reg0 <= (others => '0');
   else
    case slv_reg_write_sel is
     when "10" =>
      for byte_index in 0 to (C_SLV_DWIDTH/8)-1 loop
       if ( Bus2IP_BE(byte_index) = '1' ) then
        slv_reg0(byte_index*8+7 downto byte_index*8) <= Bus2IP_Data(byte_index*8+7 downto
byte_index*8);
       end if;
      end loop;
     when others => null;
    end case;
   end if;
  end if;
```

5. Click on the Top Module in the Design Hierachy Window then double click on the Synthesize – XST to synthesize the IP project.

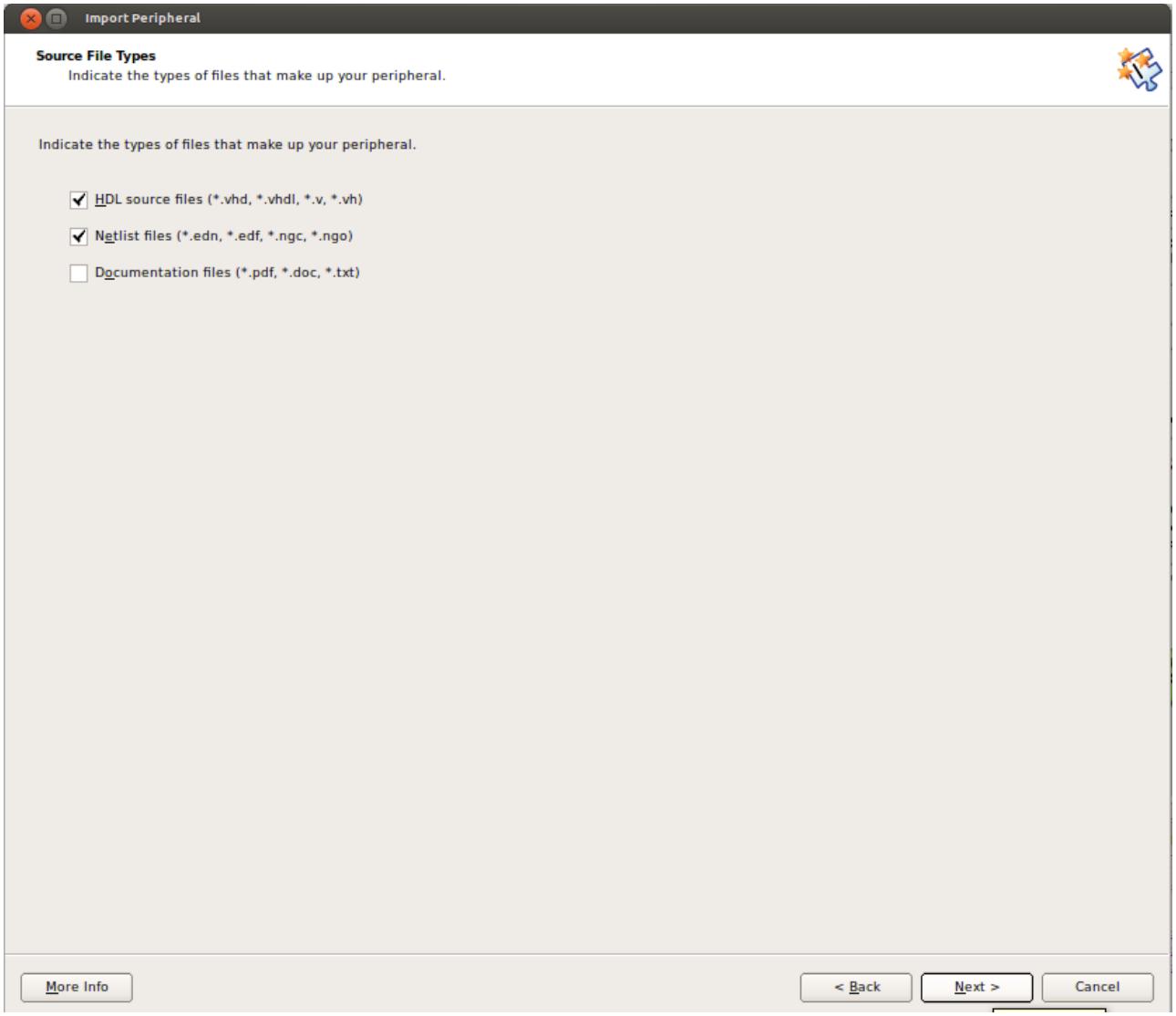
Adding the Reconfigurable Partition Black box to the system in EDK

1. In EDK, click on the Hardware tab in the toolbar and click on the Create and Import Peripheral Wizard once again. When ask to create or import a design, pick Import Existing Peripheral.

2. Click next to choose the default project to store it in.

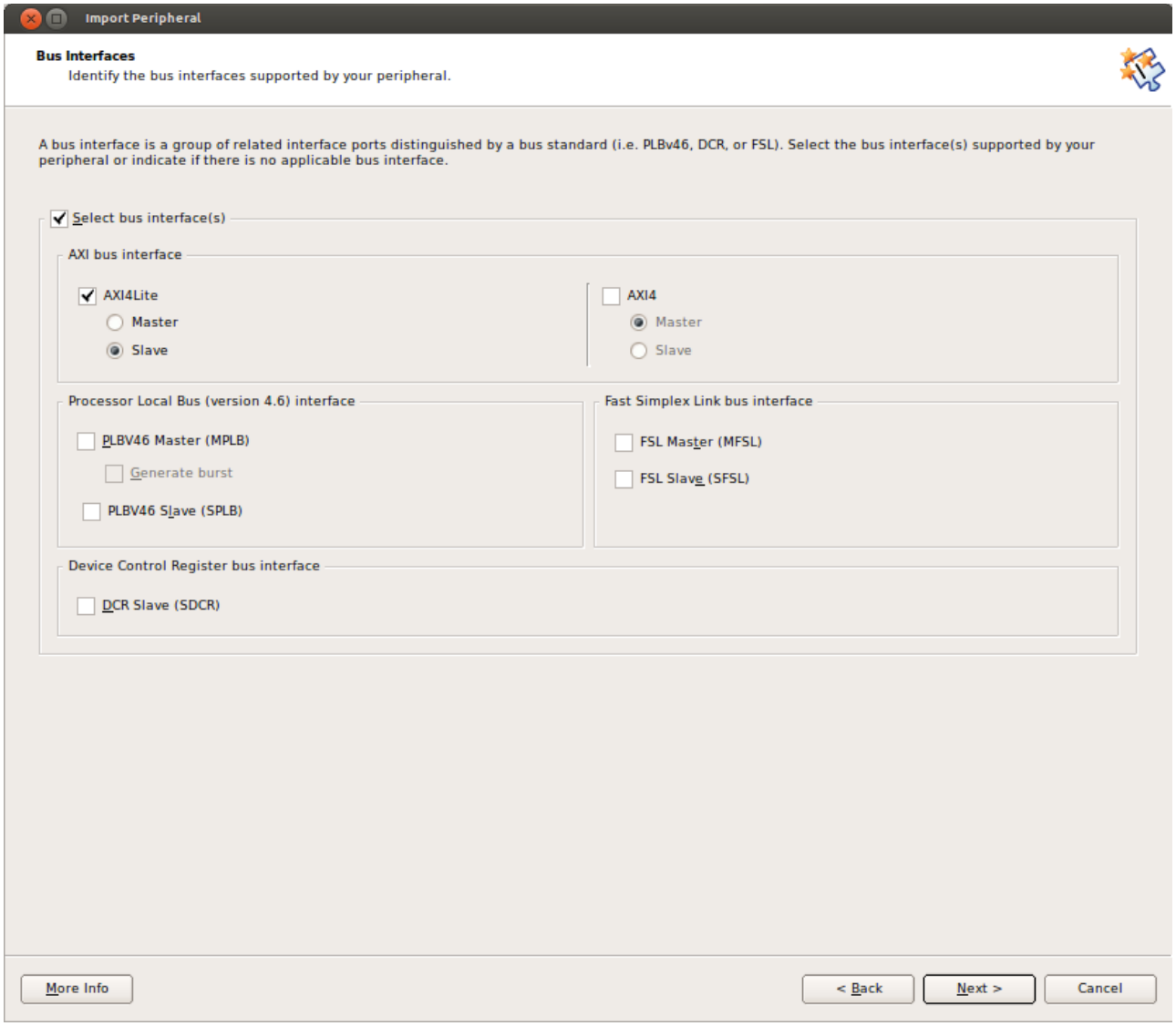
3. Name this peripheral the same name as the previous IP peripheral and overwrite it.

4. In the next window make sure that HDL source and the Netlist Files check box is clicked

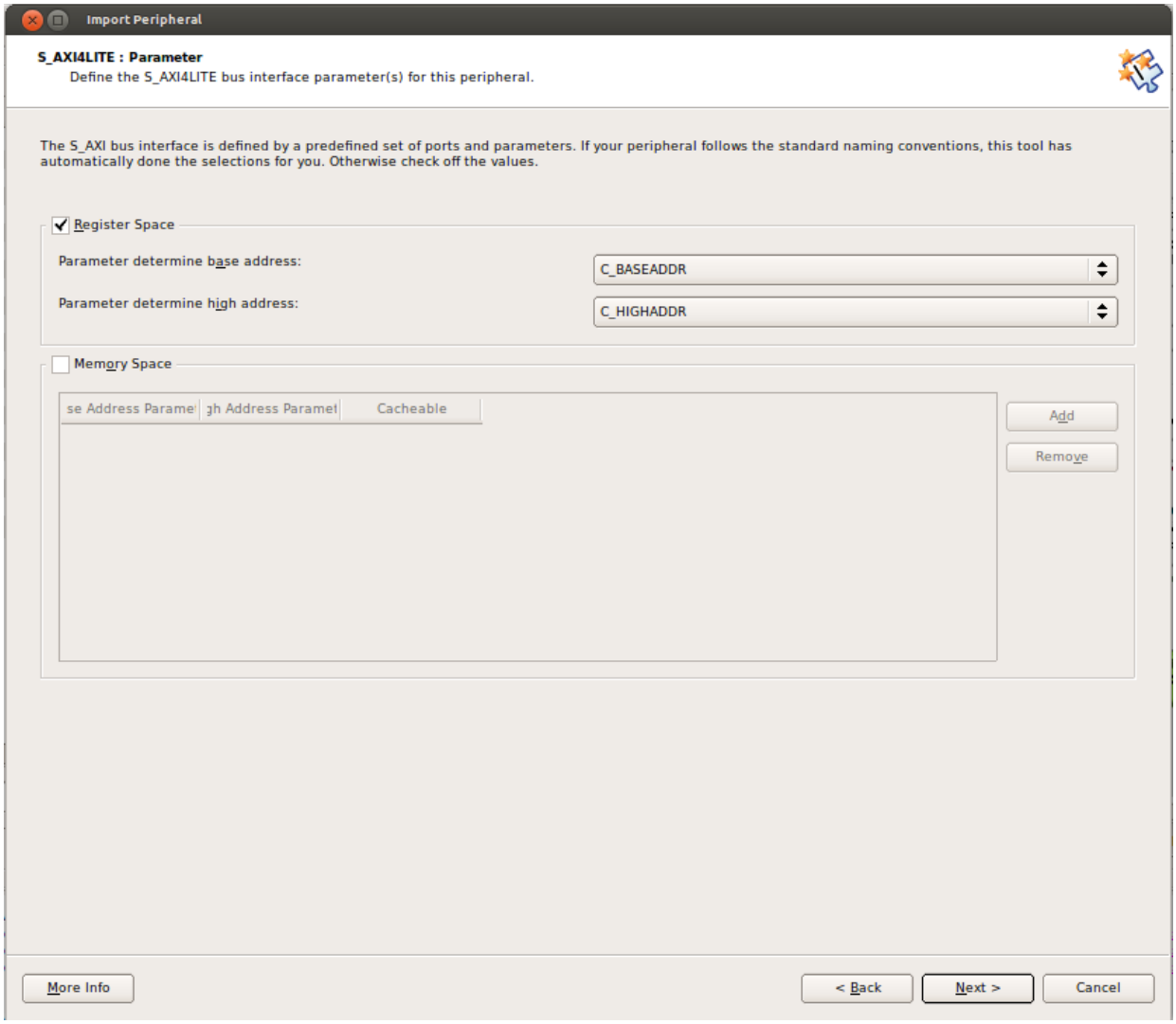


5. In the option “Use an XST project files (*prj)”, browse to the folder where the IP ISE project is located, chose the ISE project file and then click next 2 times.

6. In the Bus Interface window check the AXI4Lite and Slave option.



7. Keep clicking next until it asked to determine the High Address Parameters and under the C_BASEADDR option choose C_HIGHADDR for the High address.



8. Click until it ask to Add Netlist Files. Click Add Files and browse to your bitcountup RM folder and choose the prconfig.ngc file. Then finish the import.

Adding the IP from the repository to the system and generate an address for it.

1. In the IP Catalog window there should be a User option at the bottom. Open this option and double click the peripheral that was just made to add its instance to the system. Keep the default settings.

2. In the Address Tab in the System Assembly View window on the right side of those tab towards the end of the window, there should be a button that will generate the systems address. Click this button.

3. Now generate the Netlist by clicking Generate Netlist in the Navigator window.

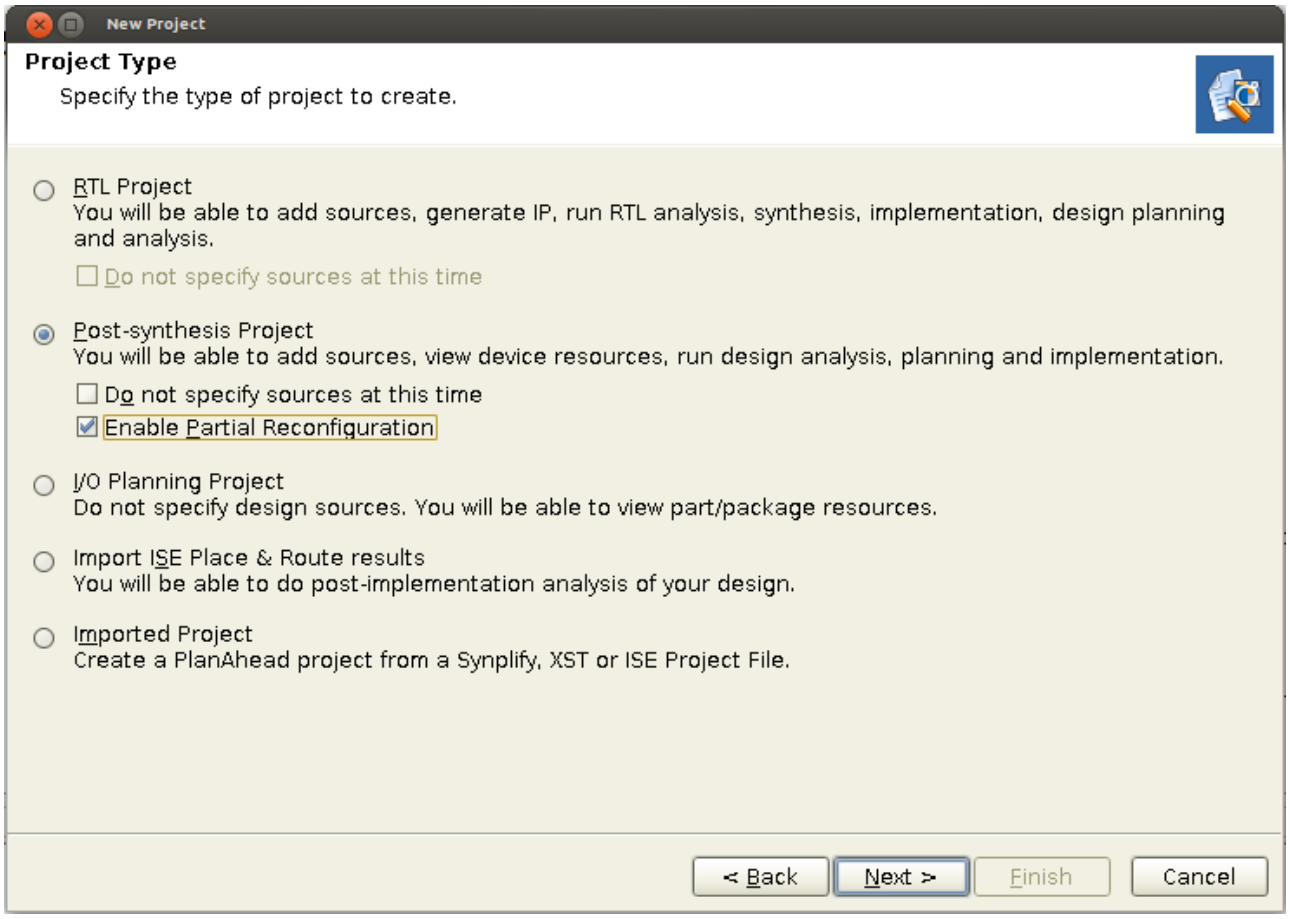
4. Once the Netlist is finish generate the bitstream and then export to SDK including the Bitstream

and BMM. Launching the SDK is not necessary at this point.

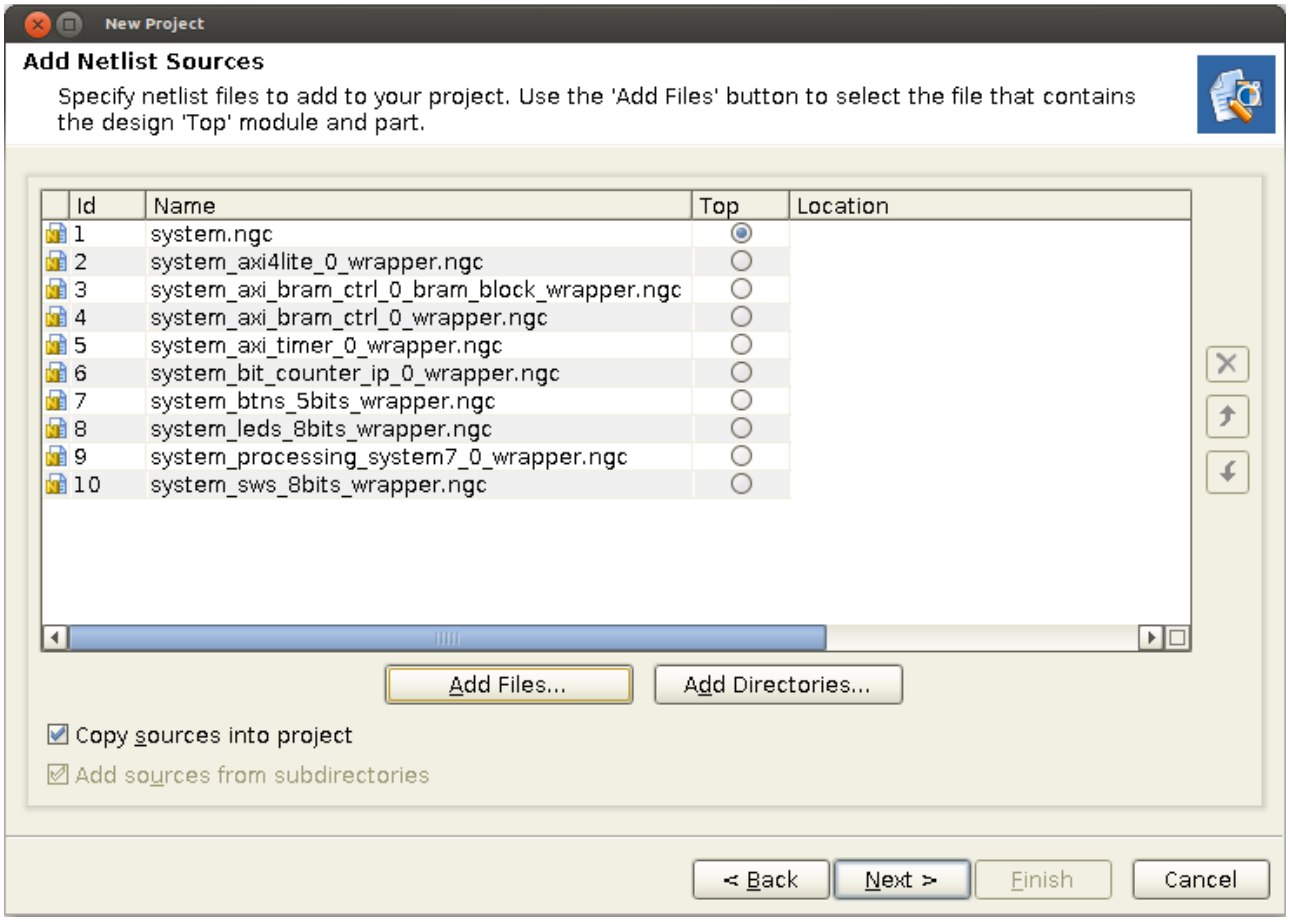
Create a PlanAhead project and Create a floorplan for the Reconfigurable Partition.

1. Create new project with the name planahead in the project folder so that the project will be built inside the planahead folder that was created earlier.

2. Click on Post-Synthesis Project, make sure “Enable Partial Reconfiguration” is checked

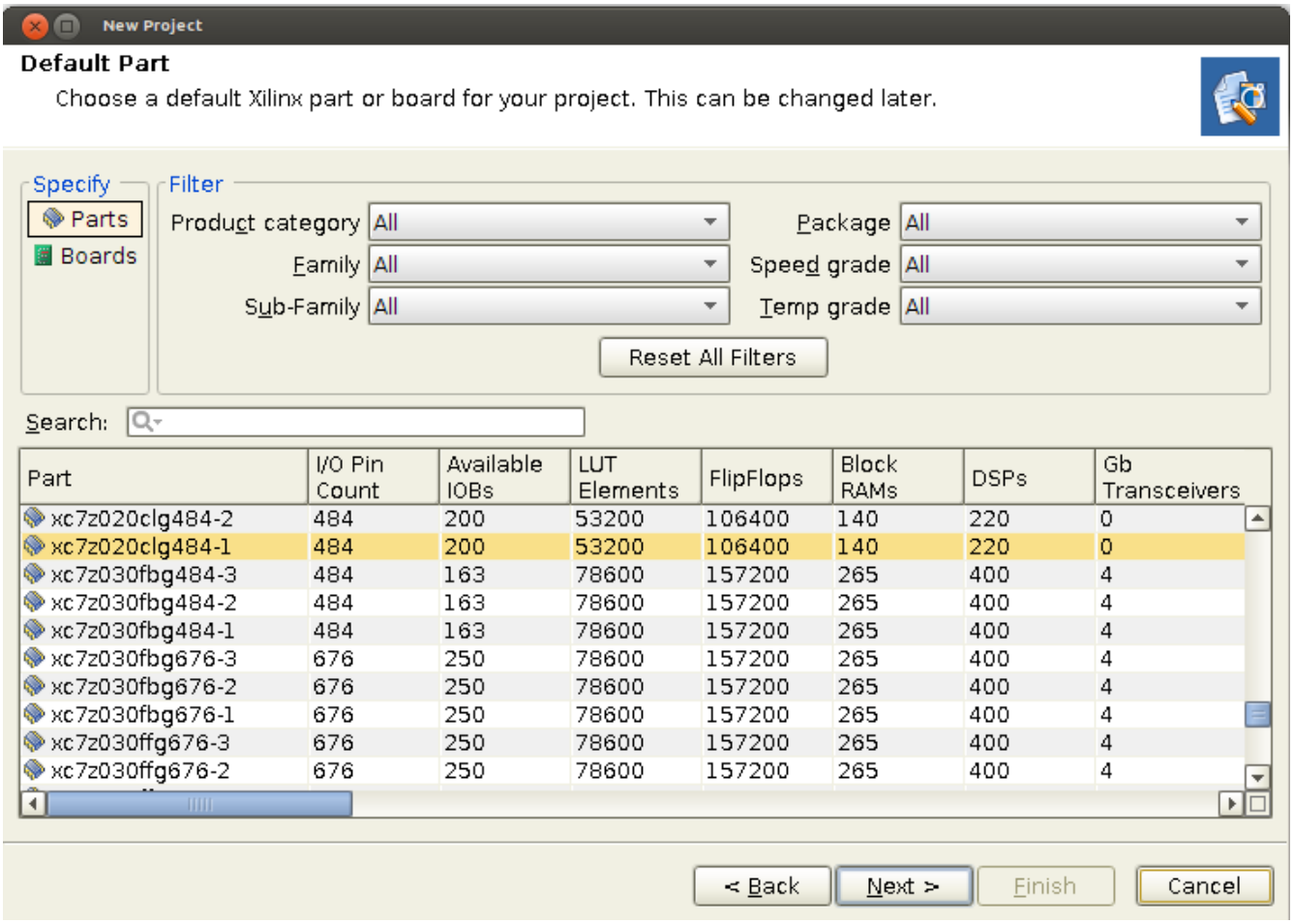


3. In the next window for adding the Netlist click on Add Files and browse to edk/implementation folder and select all the .ngc files and click okay. Before clicking next make sure that system.ngc has the Top attribute clicked then click next.



4. In the Add constraint window click Add Files, browse to edk/implementation if it is not already there and choose all the .ncf, .ucf, and .xcf files if available click okay then click next.

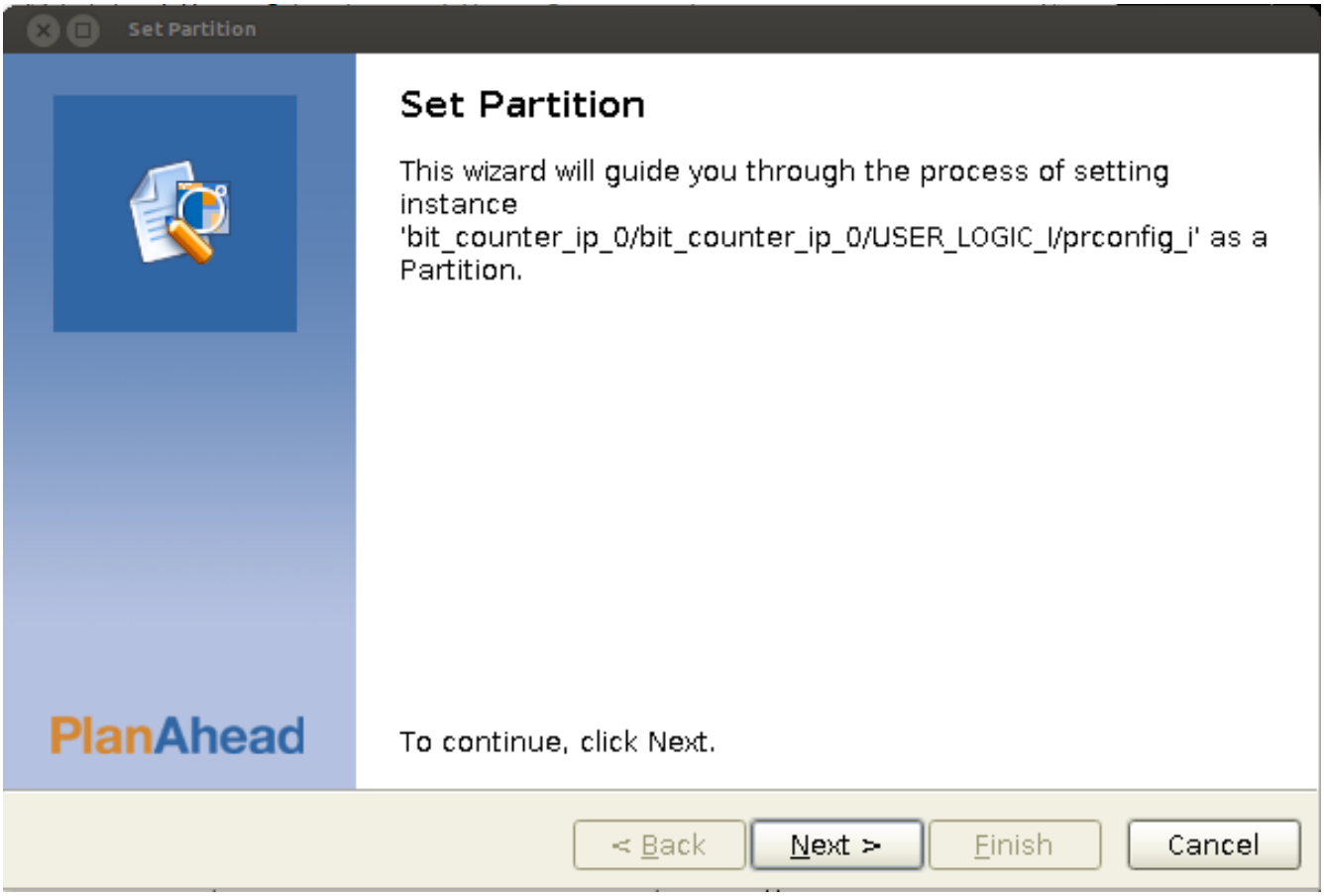
5. In the Parts Select window make sure that the xc7z030clg484-1 is selected and then Finish the setup.



Creating the Reconfigurable Partition

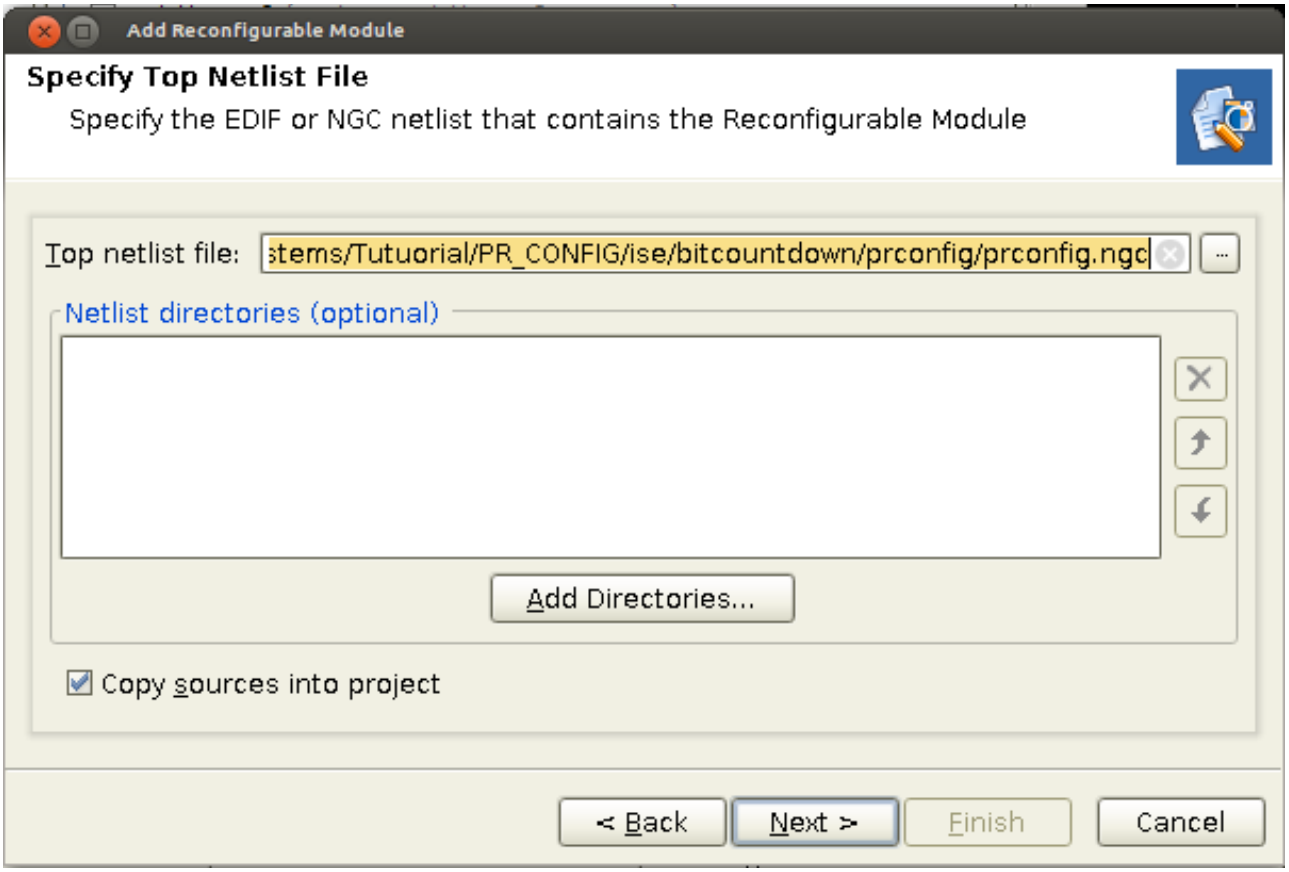
1. Whenever the project is finish building click on Open Synthesized Design in the project manager to open the synthesize system.

2. In the Netlist window the IP peripheral that was created earlier should be in the list. Click the peripheral, there should be an instance of the USER_LOGIC_I. (bit_counter_ip_0/USER_LOGIC_I/prconfig_i). Right click this instance and click Set Partition.



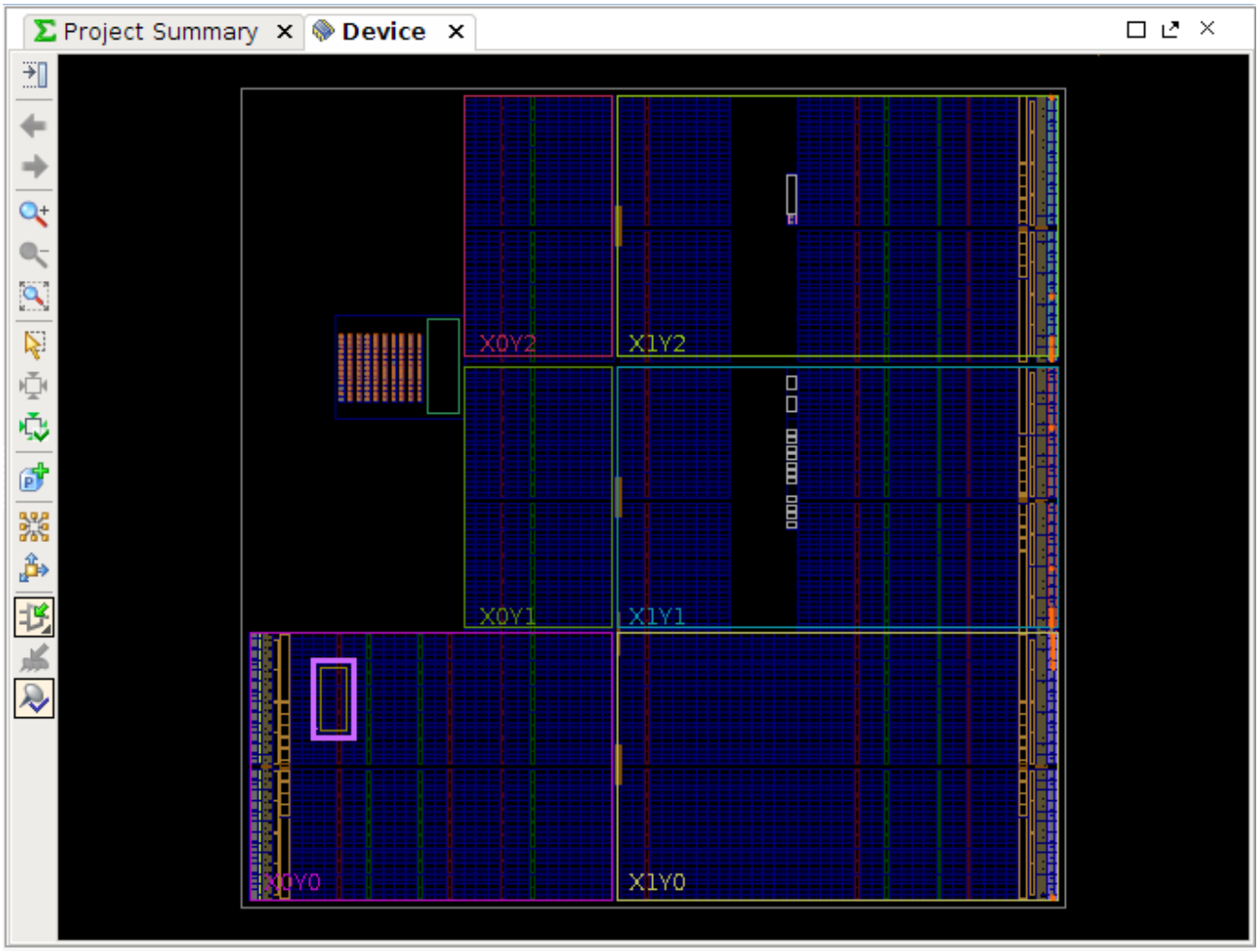
3. When the window opens click next 2 times for the default settings. There should be a dialog that says that this instance already has an implementation loaded. This just means that the Program will start with the RM that was added earlier during the creating of the IP peripheral. Name the module countup then click next until it finish.

4. Repeat the previous step for the second module with the name count down. When it ask to add the Netlist file for this module, click the browse button (…) next to the "Top Netlist File". Browse to the second RM ISE folder bitcountdown and load the prconfig.ngc file. Then click next until it finish.

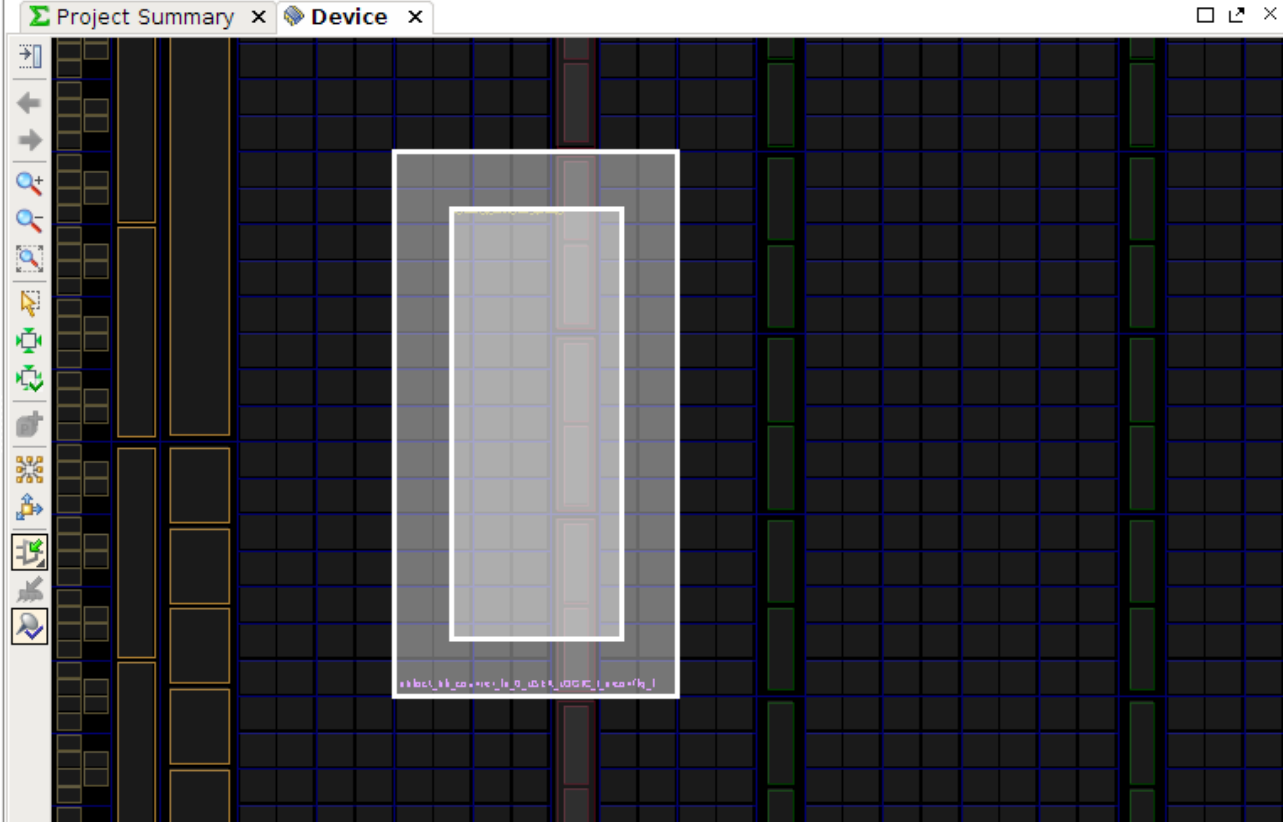


Defining the Reconfigurable Partition Region

1. Right click on the IP instance and click Set Pblock size. In the Pblock properties there should be a Physical Resource Estimate window with red text. This information determines the required resources that needs to be included. In the Device Block window draw a square around the resources that is required.



2. If all the required resources is included then in the Physical Resource Estimate window there should be a Utilized tab. In this tab make sure that it is not close to 100%. If it is then there is a chance that there are not enough resources for the other RM.



3. Now click on the Tools tab on the toolbar and click Report DRC to check on the Design rule for the Partial Reconfiguration. Make sure that Partial Reconfiguration is the only one that has a check next to it. Ignore the warning that the RM are not implemented. It will be implemented later.

Implementing the First Configuration, Implementing, and Promoting

1. Click on config_1 in the Design Runs window and change the name to countup in the General Tab of the Implementation Run Properties Window. Make sure the strategy is set for ISE Defaults in the Option tab.

2. Right click on countup in the Design Runs window and click Launch Runs. Run on the local host and click next. After the launch is finish a window will pop up, click the “Promote Partition” the click okay.

3. Right click on the second RM in the Netlist and Set it to Active.

4. Right click on the Design Run window and Create Runs.

5. Rename it to countdown and click next and run it on local host.

6. Once the launch is finish click on “Verify Configuration” under the Program and Debug. Select all the RM when the window appears and click okay.

Generating the Bitstream and Binary file.

1. Highlight all the runs in the Design Run, right click and Generate the bit stream.

2. In order to generate the Binary file that will be copied to the SD card so that the FPGA can transfer the partial Bitstream into the DDR memory. It must be done in the Tcl Console in Planahead. This next step will convert the partial bit stream to a binary file and move it to the bit stream folder that was created earlier.

3. In the Tcl console enter:

```
exec promgen -b -w -p bin -data_width 32 -u 0
planahead.runs/bitcountup/bitcountup_bit_counter_ip_0_bit_counter_ip_0_USER_LOGIC_I_prco
nfig_i_bitcountup_partial.bit -o ../bitstreams/bitcountup.bin
```

4. In the console you will see some information about the binary files:

```
Release 14.7 - Promgen P.20131013 (lin64)
Copyright (c) 1995-2013 Xilinx, Inc.  All rights reserved.
0x1a650 (108112) bytes loaded up from 0x0
Using generated prom size of 128K
Writing file "../bitstreams/bitcountup.bin".
Writing file "../bitstreams/bitcountup.prm".
Writing file "../bitstreams/bitcountup.cfi".
INFO:Bitstream:182 - The start address provided has been multiplied by a factor
  of 4 due to the use of the x32 data width
```

5. Pay Attention to the highlighted number. This is the size of the bit stream.

6. Rename the generated bin files from countup.bin to bcu.bin and countdown.bin to bcd.bin. The reason for this is that when the Software is trying to load the bit stream to the DDR, a longer name may give the SD card reader issues of reading the file.

7. Load all the bin files to your SD card.

8. In the first module runs folder planahead.runs/countup/ there should be a bit file with the name countup.bit. This will be your full bit file that will be programed to the FPGA in SDK. Rename this file download.bit.

Software Implementation thought SDK

1. Open SDK and switch your workspace to the edk/SDK/SDK_Export folder where your hardware was exported to.

2. Click File>New>Project...>Xilinx>Hardware Platform Specification. To build a HW platform.

March 2015

3. Give it a name and browse to the hw folder in SDK_export and click the .xml file.

4. Click File>New>Project...>Application Project to create a new project. Make sure the Hardware Platform is the one you just made in the previous step and click finish.

5. Copy the Sample code from the provided Files.

6. Program the Zedboard using the download.bit file that was made earlier then run the project on the board.

7. You should see your LEDs count up in binary. When you click any of the buttons the board should reconfigure from the RM bit file, the LEDs should stop and start to count down. Every time one of the buttons is pushed, the FPGA will reconfigure between the 2 RM.

References

"Digilent Inc. - Digital Design Engineer's Source." *Digilent Inc. - Digital Design Engineer's Source*. N.p., n.d. Web. 25 Mar. 2015. <http://www.digilentinc.com/>.

Xilinx, Inc. *Xilinx Partial Reconfiguration of a Processor Peripheral Tutorial: PlanAhead Design Tool* (n.d.): n. pag. *Www.xilinx.com*. Xilinx, 25 Apr. 2013. Web. 25 Mar. 2015. <http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/PlanAhead_Tutorial_Reconfigurable_Processor.pdf>.

"Zedboard." *Zedboard*. N.p., n.d. Web. 25 Mar. 2015. <http://www.zedboard.org/>.

"Zynq 7000 Partial Reconfiguration Reference Design." *Xilinx Wiki* -. Xilinx, n.d. Web. 25 Mar. 2015. <http://www.wiki.xilinx.com/Zynq+7000+Partial+Reconfiguration+Reference+Design>.