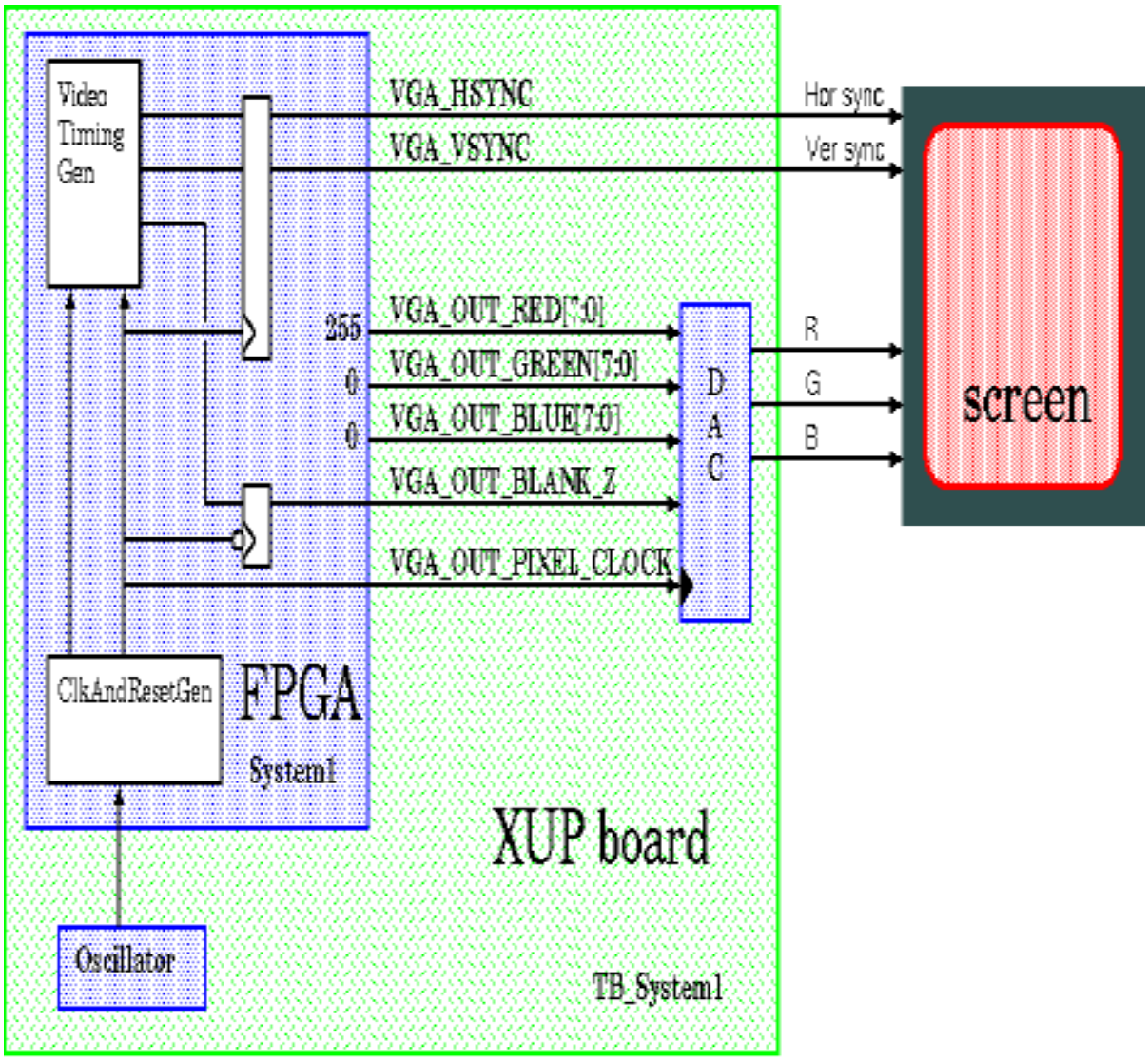
Lab-exercise

Lab6:
Implementation of a video timing generator

Introduction

In the previous modules of this lab you built and verified a system with a VHDL simulator. In this module you are going to expand this system and eventually load it in to the FPGA and be able to admire your work on the screen.



Objectives

After having done this lab, the student should understand the tool flow enough to describe a hardware system like the video timing generator in VHDL, verify it by simulation, synthesize it and load it in an FPGA.

Knowledge background

Modules 1a, 1b en 1c should first be finished.

Classification

Difficulty on a scale of 1 tot 5: 1
Time needed (without support): 1.5 to 2 hours

Input

The following folders/files are in the module2a folder:

- Hardware : all files to build System1 (cf. previous assignments)
- Simulation : all files to simulate System1
- Implementation: constraints file to implement System1

Using this structure allows for easier re-use of parts of this design in other ones.

The lab

1. Pixel clock and reset generation for the Video timing generator.

On the XUP board a crystal generates a 100MHz clock that is connected to the FGPA. This
clock can not directly be used as pixel clock because the pixel clock should run at 25Mhz. The crystal clock has hence to be divided by 4. The easiest and best way to do this on a Xilinx VirtexII-Pro is using a digital clock manager (DCM). The FPGA on the XUP board has eight of these units. A DCM can be used to divide clocks, multiply them and shift their phase. Internally the DCM units have a delay locked loop (DLL) making it possible to eliminate the skew of the output clock with respect to the input clock totally. These DCM units however need a start-up time during which the output clock(s) do not have the correct frequency and/or phase. During this start-up time their “locked” output remains low. There’s more information in the Virtex-II Pro Platform FPGA User Guide. To keep things simple for a student, a ClkAndResetGen entity is made available. This entity uses a DCM to generate the pixel clock and the asynchronous reset.

```
ENTITY ClkAndResetGen IS
generic(
Div_factor : real := 4.0
```

```
);
port(
SYSTEM_CLOCK : in std_logic; -- 100 MHz clock
Clk : out std_logic; -- 50/25 MHz clock
Reset_n : out std_logic -- active low reset
);
end ClkAndResetGen;
```

The simplified architecture looks like this:

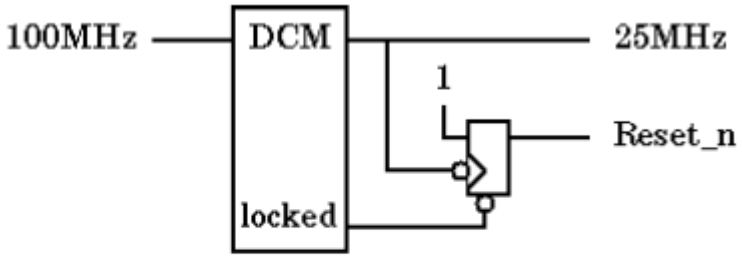


Figure 2

Question 1 : Why the extra flip-flop? ?

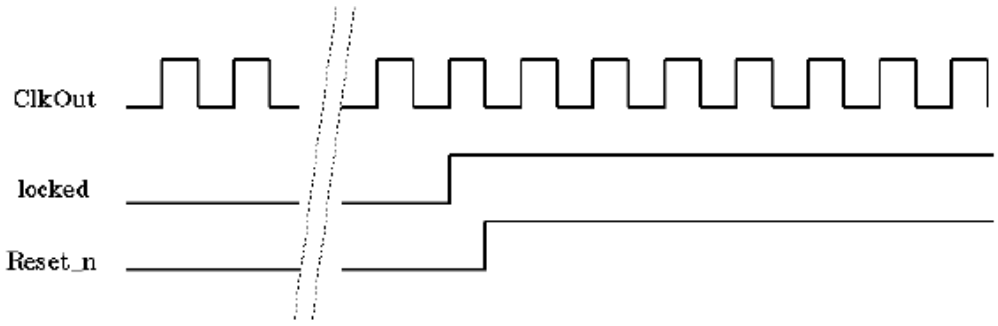


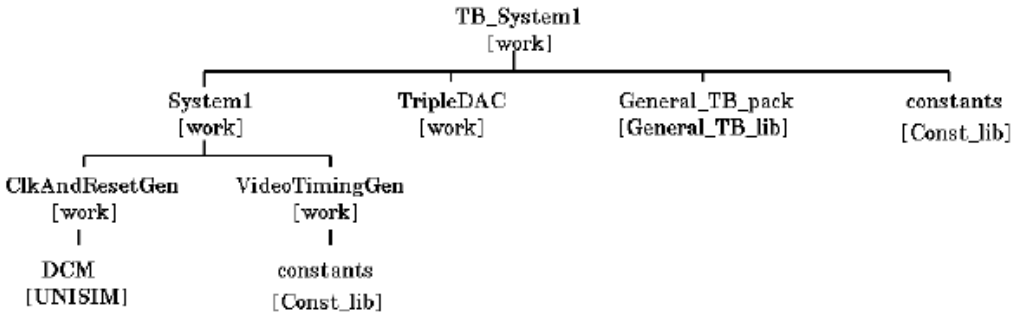
Figure 3

2. Putting together the complete system "System1"

In System1.vhd the complete system as described in the introduction to this module is put together. As you can see in the VHDL model of the digital to analog converter, this DAC introduces an extra clock period delay on the RGB signals. Question 2: What is the function of the flip-flops on the horizontal and vertical synchronization pulses in figure 1? Question 3:In System1 we keep the RGB values fixed, but if we would let them change, we would also have to capture them first in a register clocking on the falling edge of the pixel clock, just like the VGA_OUT_BLANK_Z signal (= video_on in the VideoTimingGen). Why are these flip-flops necessary?

3. System1 Simulation

Simulate System1 with Modelsim in the directory module2a/Simulation.
All necessary files are in the directories Hardware and Simulation. Keeping them in two different
folders keeps the testbenches and the hardware to be implemented nicely separated.
In this lab we will compile the Constants_pack in the library Const_lib and the General_TB_pack
in the library General_TB-lib. All the rest of the VHDL files are compiled in the library WORK.



Figuur 4

- `vlib work`
- `vmap work work`
- `vlib Const_lib`
- `vmap Const_lib Const_lib`
- `vlib General_TB_lib`
- `vmap General_TB_lib General_TB_lib`
- `vcom -work Const_lib ../Hardware/Constants_pack.vhd`
- `vcom ../Hardware/ClkAndResetGen.vhd`
- `vcom ../Hardware/VideoTimingGen.vhd`
- `vcom ../Hardware/System1.vhd`
- `vcom -work General_TB_lib General_TB_pack.vhd`
- `vcom tripleDac.vhd`
- `vcom TB_System1.vhd`
- `vsim -t ps work.TB_System1`

Students liking a challenge can adapt the VHDL files and the library structure such that the WORK library is not used anymore.

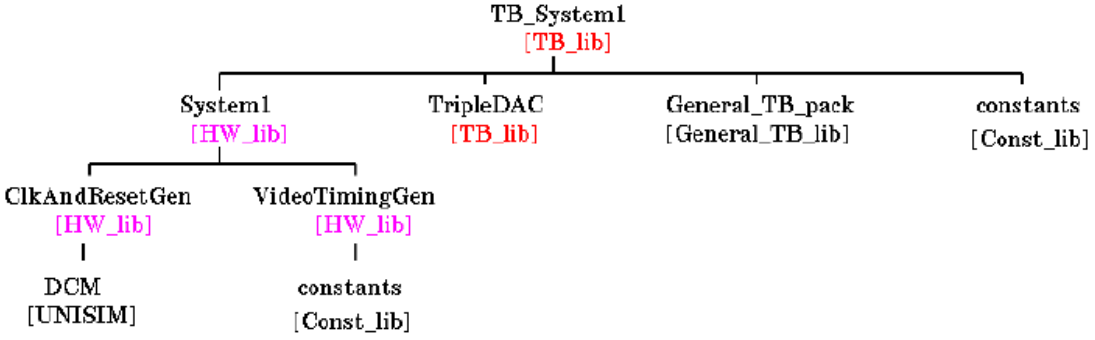


Figure 5

Now simulate the TB_System1. Make sure that the IO of System1 is visible in the waveform viewer so you can check the timing of the sync signals visually. Only at time 0 ns, you should get the following warnings.

```
# ** Warning: NUMERIC_STD.">=": metavalue detected, returning FALSE
# Time: 0 ps Iteration: 0 Instance: /tb_system1/xfpga/u2
# ** Warning: NUMERIC_STD.">=": metavalue detected, returning FALSE
# Time: 0 ps Iteration: 0 Instance: /tb_system1/xfpga/u2
# ** Warning: NUMERIC_STD."=": metavalue detected, returning FALSE
# Time: 0 ps Iteration: 0 Instance: /tb_system1/xfpga/u2
# ** Warning: NUMERIC_STD."<": metavalue detected, returning FALSE
# Time: 0 ps Iteration: 0 Instance: /tb_system1/xfpga/u2
# ** Warning: NUMERIC_STD."<": metavalue detected, returning FALSE
# Time: 0 ps Iteration: 0 Instance: /tb_system1/xfpga/u2
```

4. Implementation of System1

The file System1.ucf contains all constraints the synthesis and place & route tools have to take
into account. These are:

- The SYSTEM_CLOCK period. On the board there is a 100 MHz crystal, hence

```
TIMESPEC "TS_SYSTEM_CLOCK" = PERIOD "SYSTEM_CLOCK" 10.00 ns HIGH 50 %;
```

- The pin positions of the FPGA:

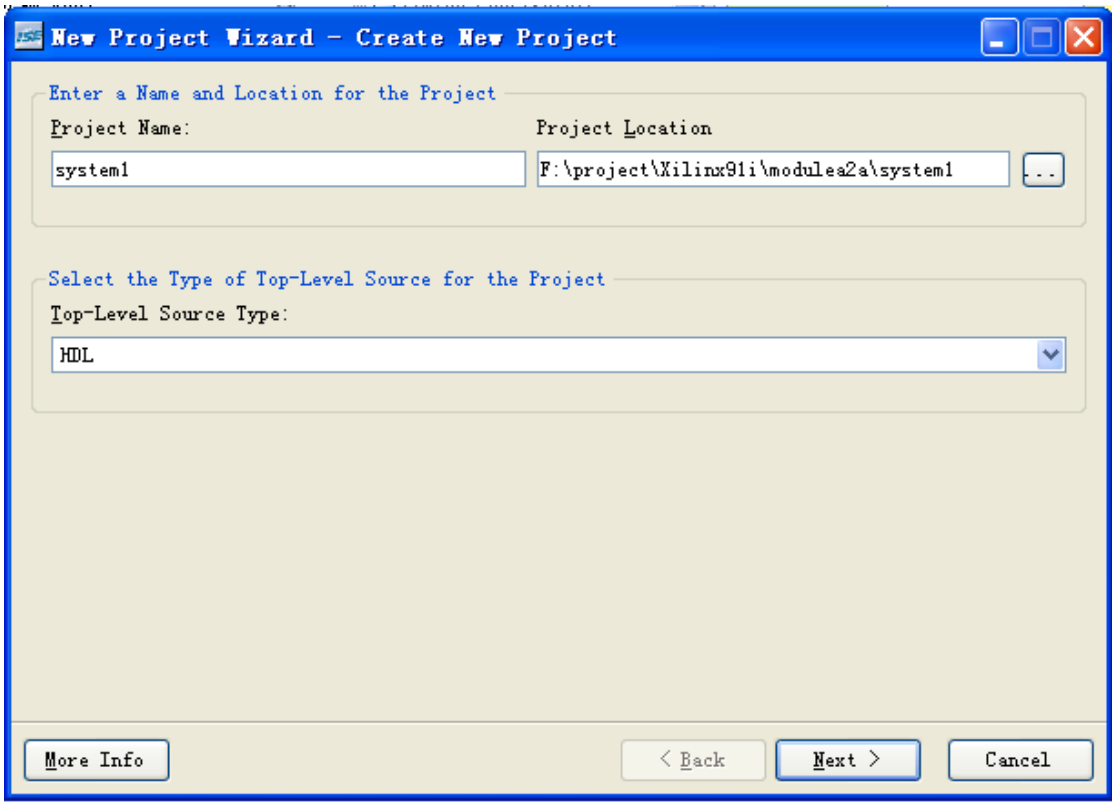
```
NET "SYSTEM_CLOCK" LOC = "AJ15";
NET "VGA_VSYNCH" LOC = "D11";
```

- The configuration of the IOs

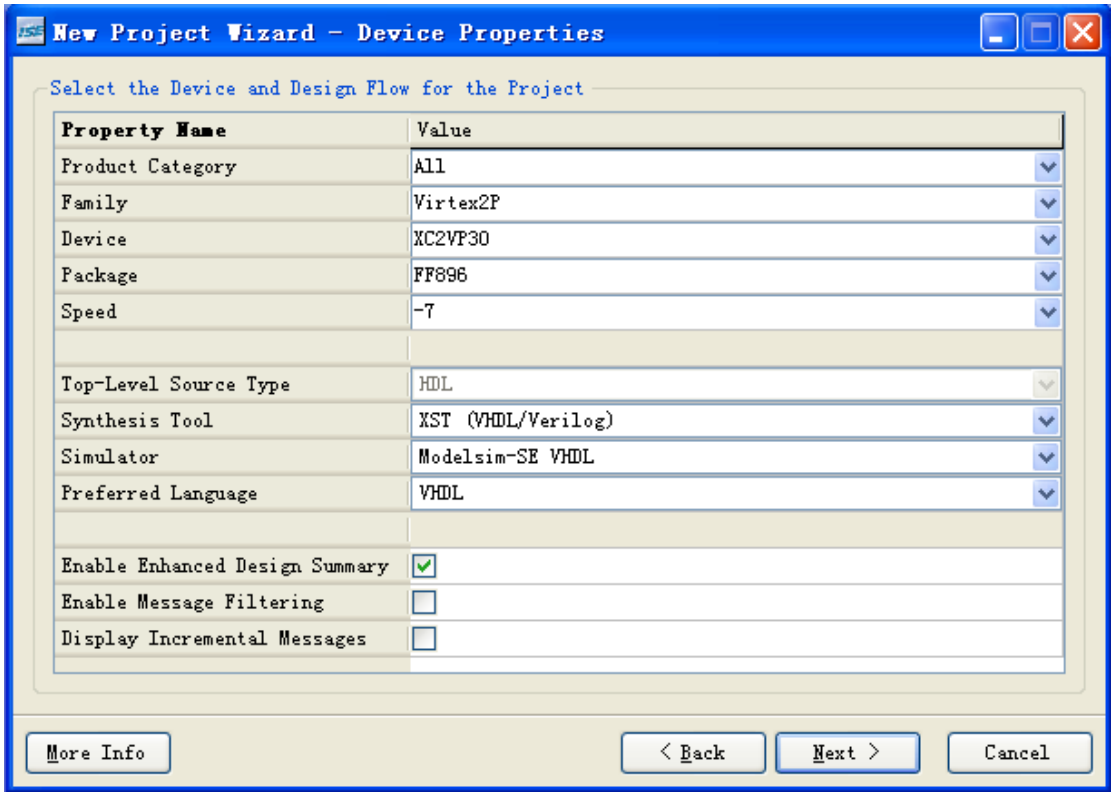
```
NET "VGA_VSYNCH" IOSTANDARD = LVTTL;
NET "VGA_VSYNCH" DRIVE = 12;
NET "VGA_VSYNCH" SLEW = SLOW;
….
```

Start the ISE project navigator

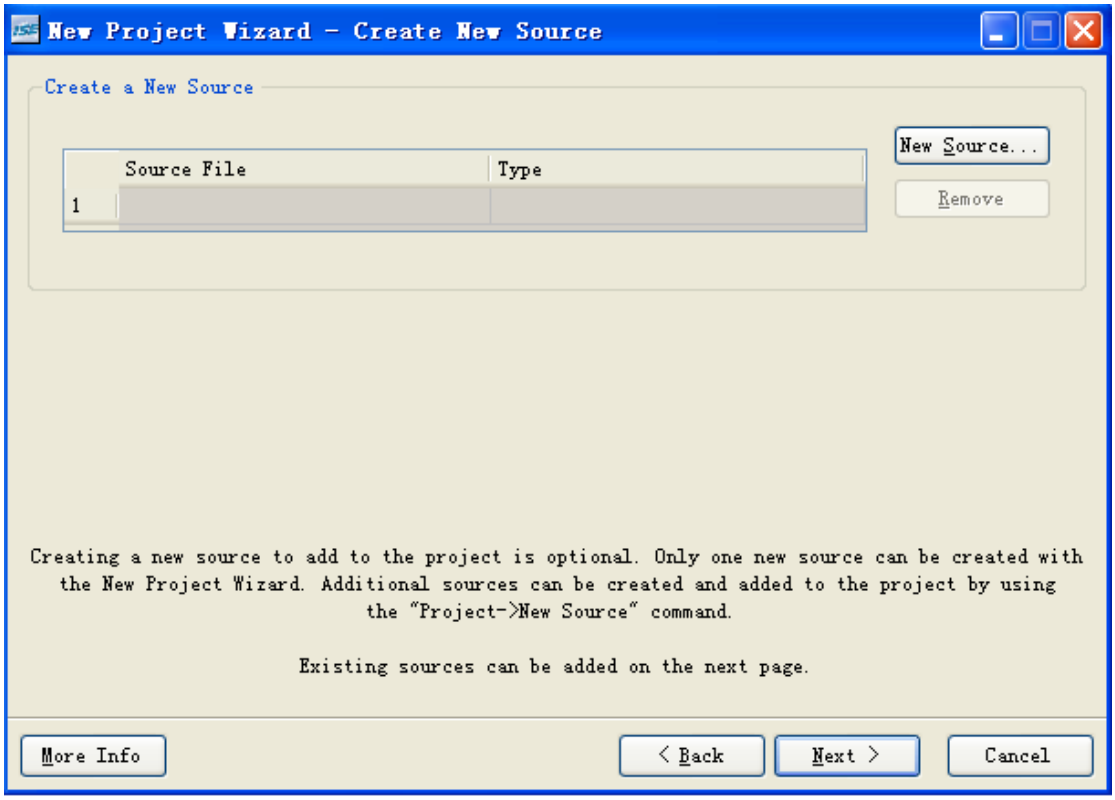
1. Open a new project in the directory module2a/Implementation. Use System1 as project name.

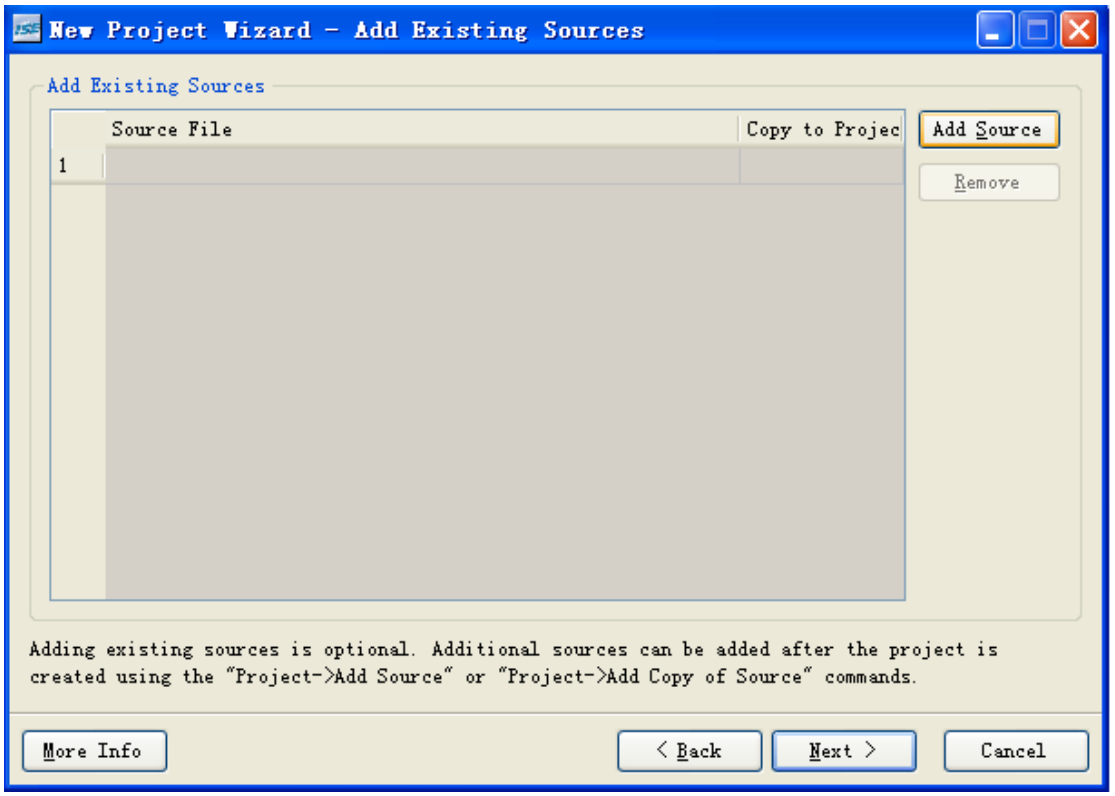


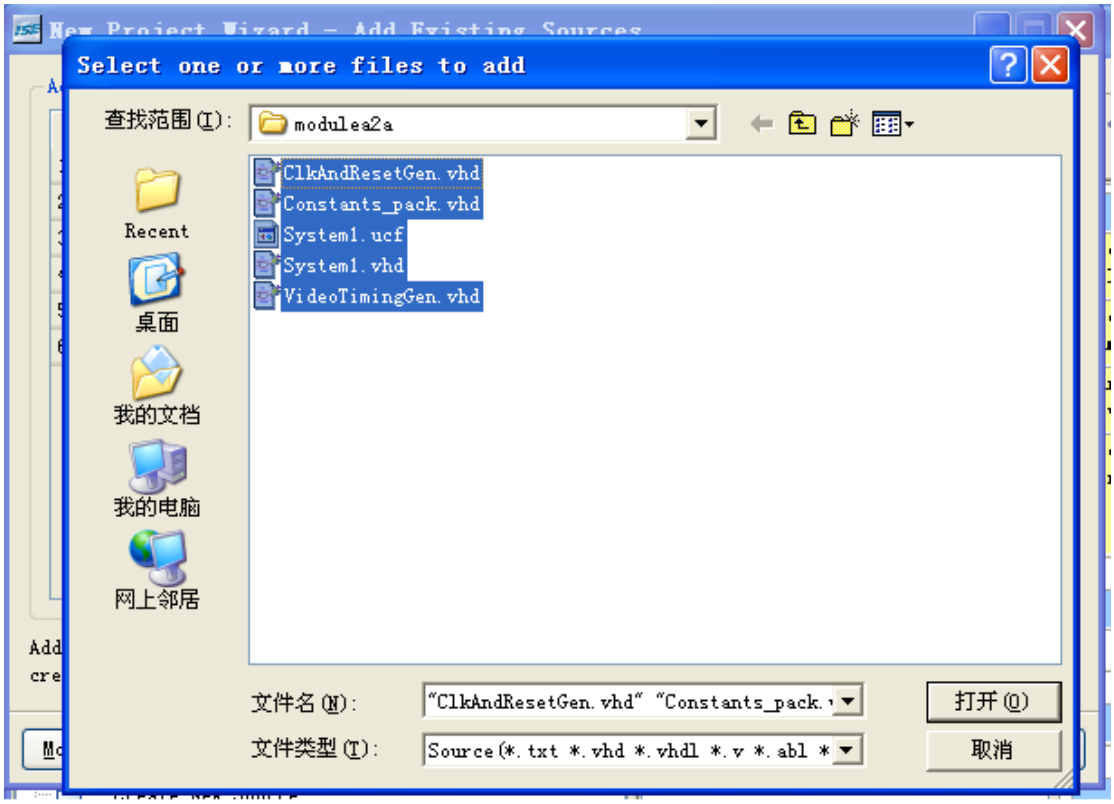
2. Choose the correct FPGA

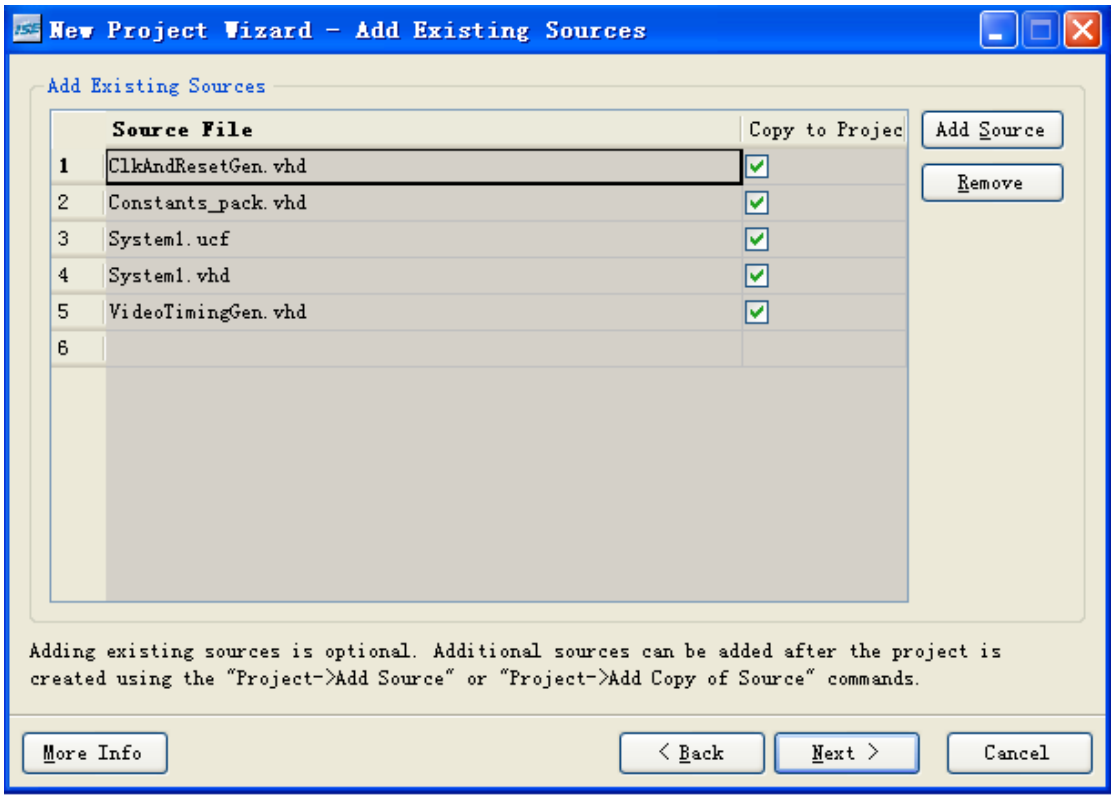


3. Add the existing VHDL files. I.e. select Next; do not select New Source.


New Project Wizard - Create New Source
Create a New Source
Source File
Type
1
New Source...
Remove
Creating a new source to add to the project is optional. Only one new source can be created with the New Project Wizard. Additional sources can be created and added to the project by using the "Project->New Source" command.
Existing sources can be added on the next page.
More Info
< Back
Next >
Cancel

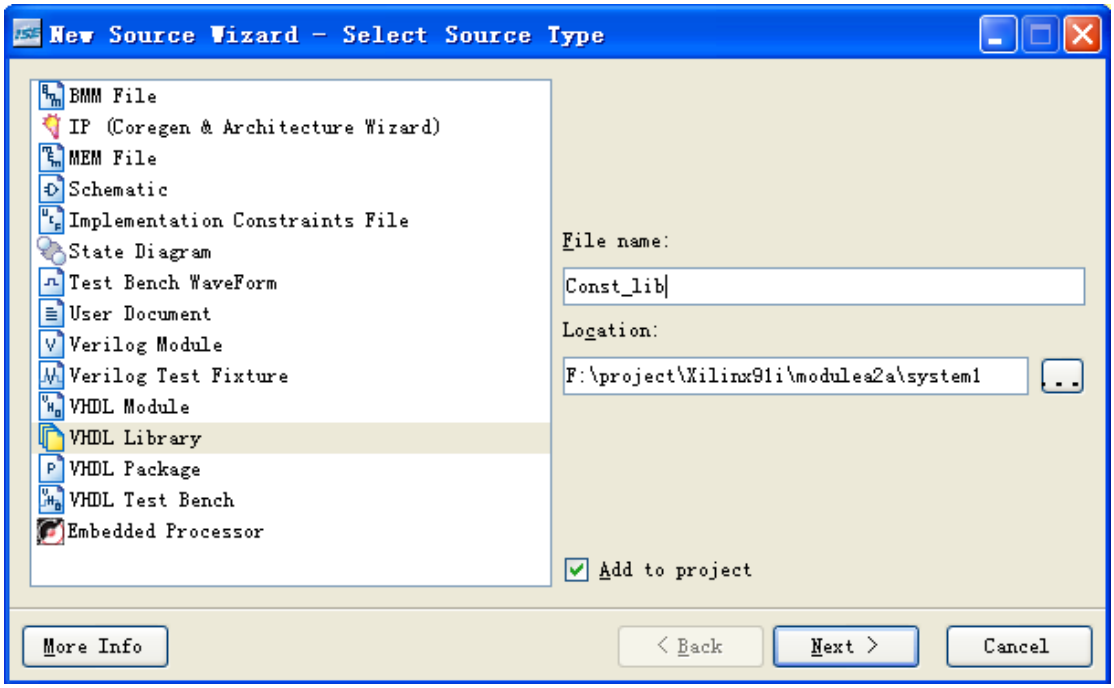

New Project Wizard - Add Existing Sources
Add Existing Sources
Source File
Copy to Projec
1
Add Source
Remove
Adding existing sources is optional. Additional sources can be added after the project is created using the "Project->Add Source" or "Project->Add Copy of Source" commands.
More Info
< Back
Next >
Cancel



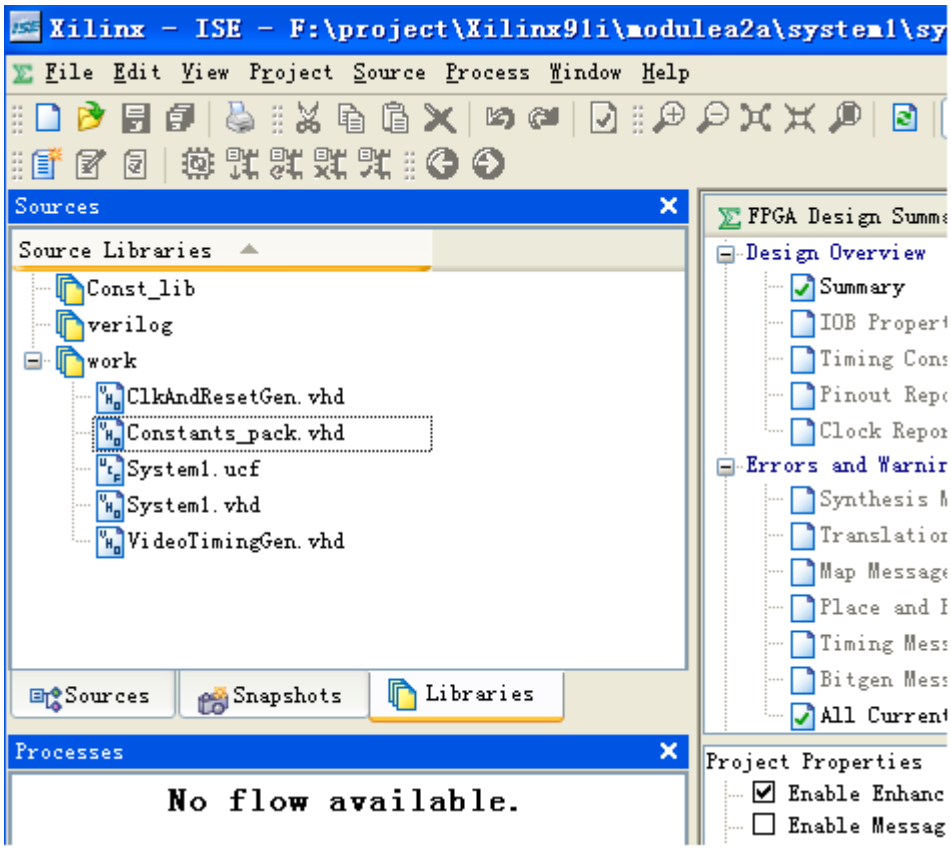


Then Click Next,
Then Click Finish,
Then Click Ok.

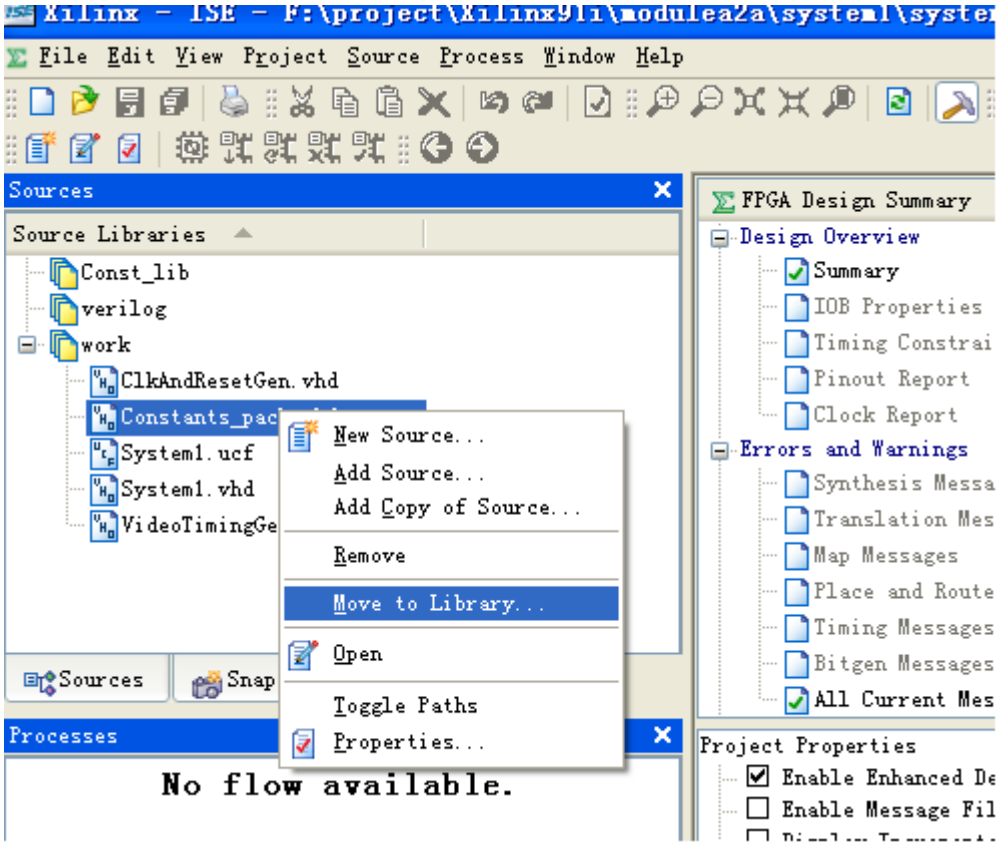
4. Create the Const_lib VHDL library: Project new source



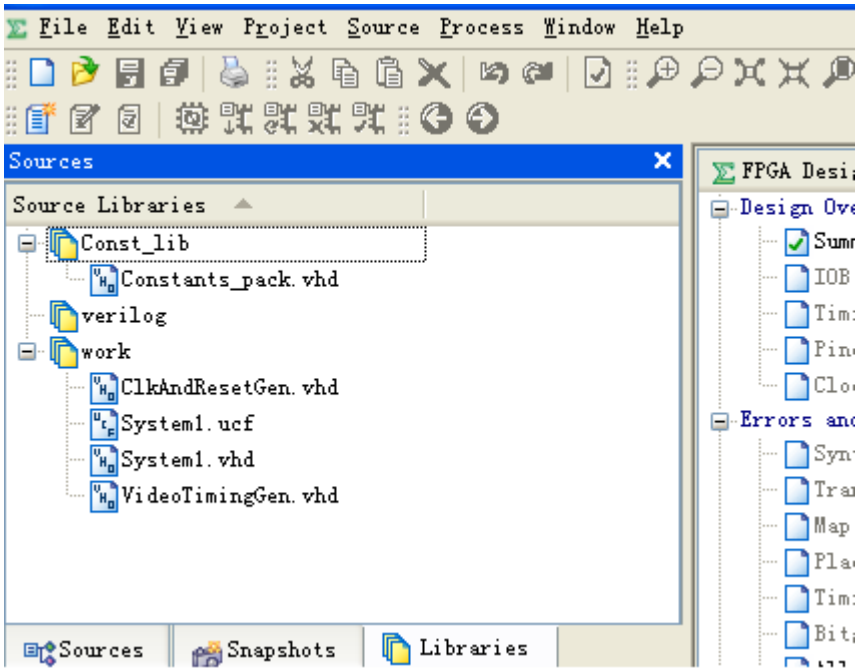
Then click Next,
Then click Finish.



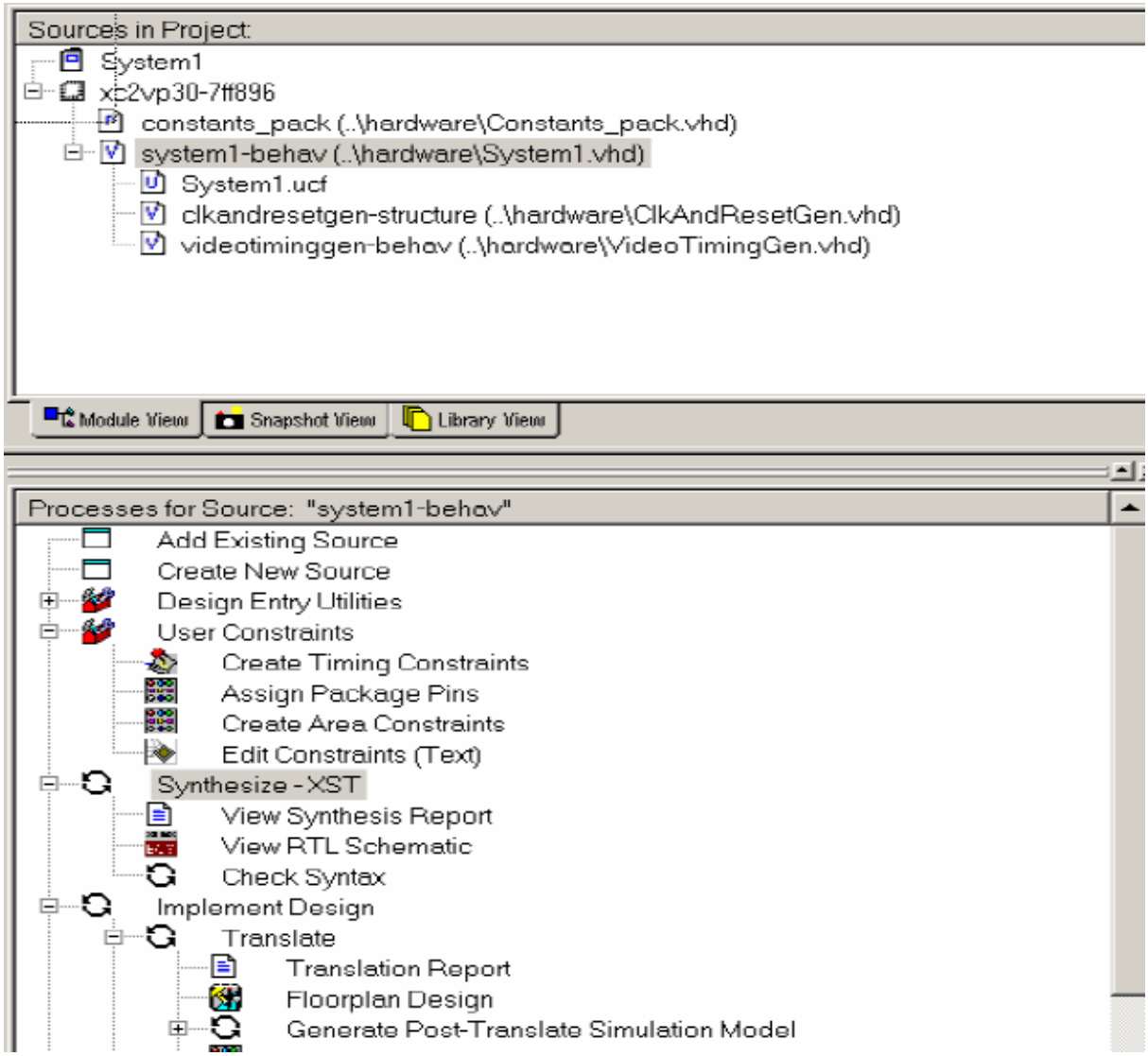
5. Right click Constants_pack.vhd and select Move to library



Then click Ok



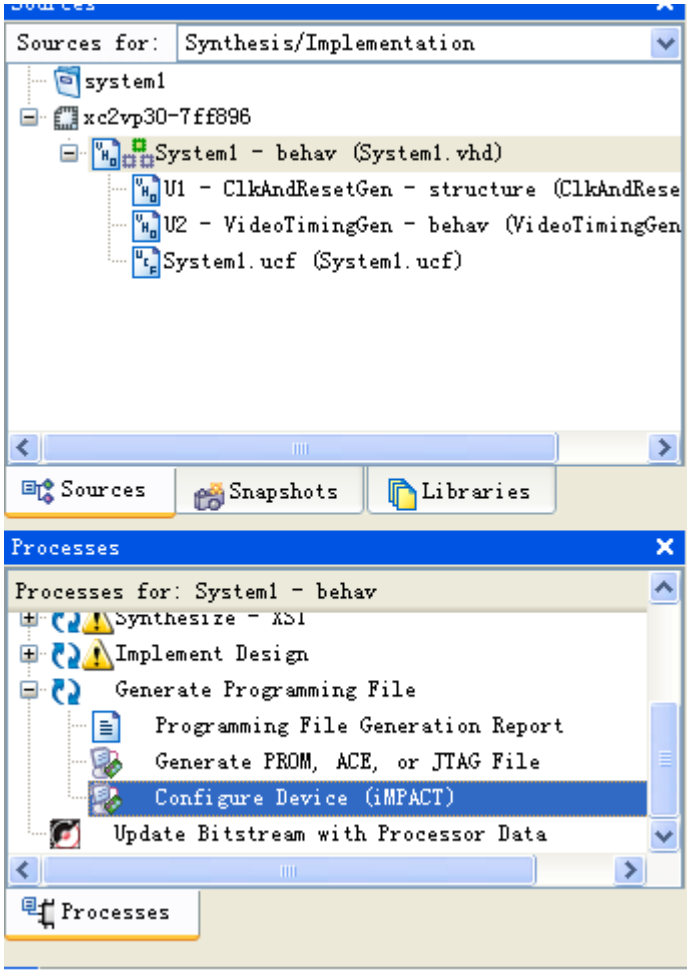
6. Start the synthesis of system1



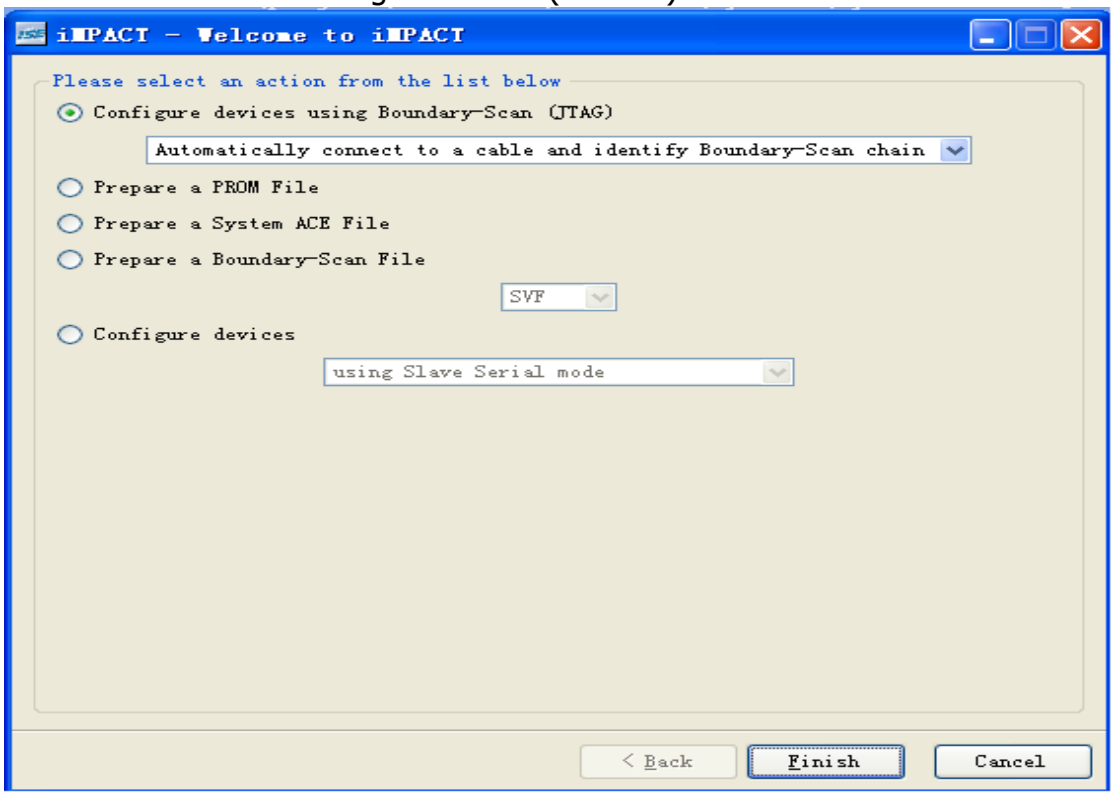
7. Implement system1 (Place en route)

8. Generate the bit (download) file

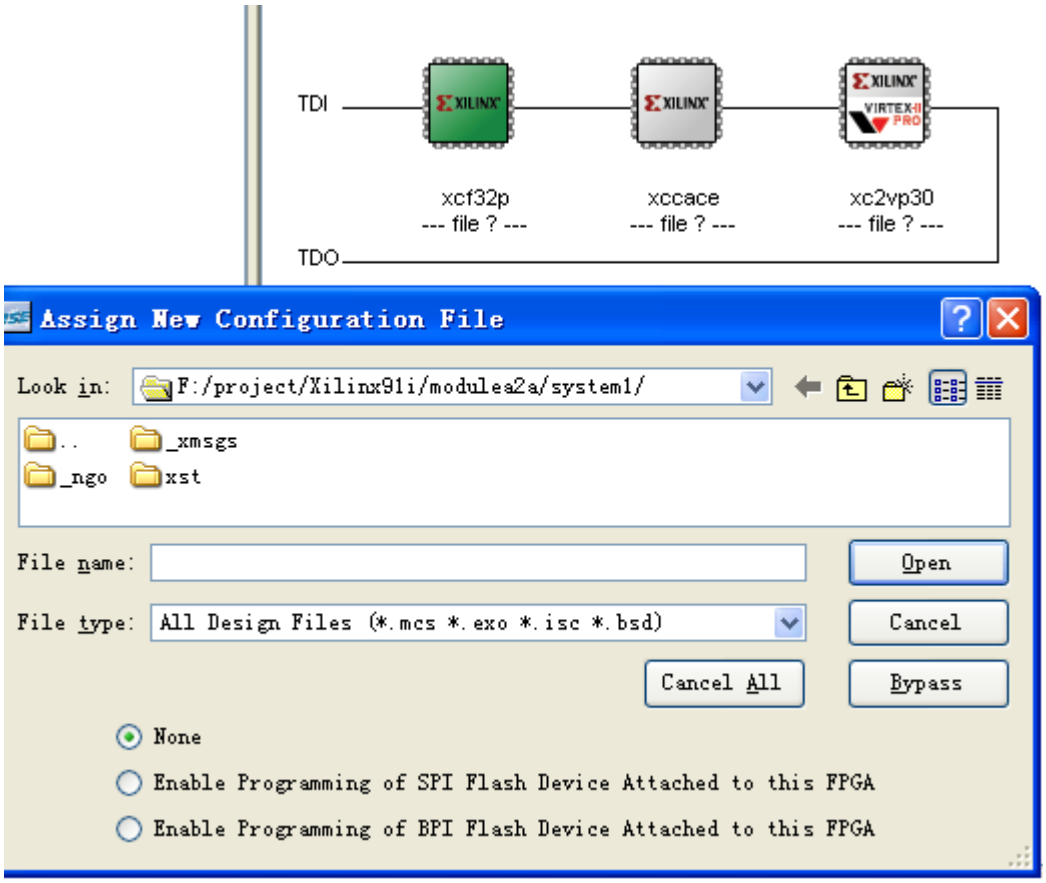
9. Connect the XUP board.(power switch = OFF!) Connect power, VGA and USB connectors Make sure that the "config source" dip switch (SW9 on the XUP)is set to JTAG Switch on the power.



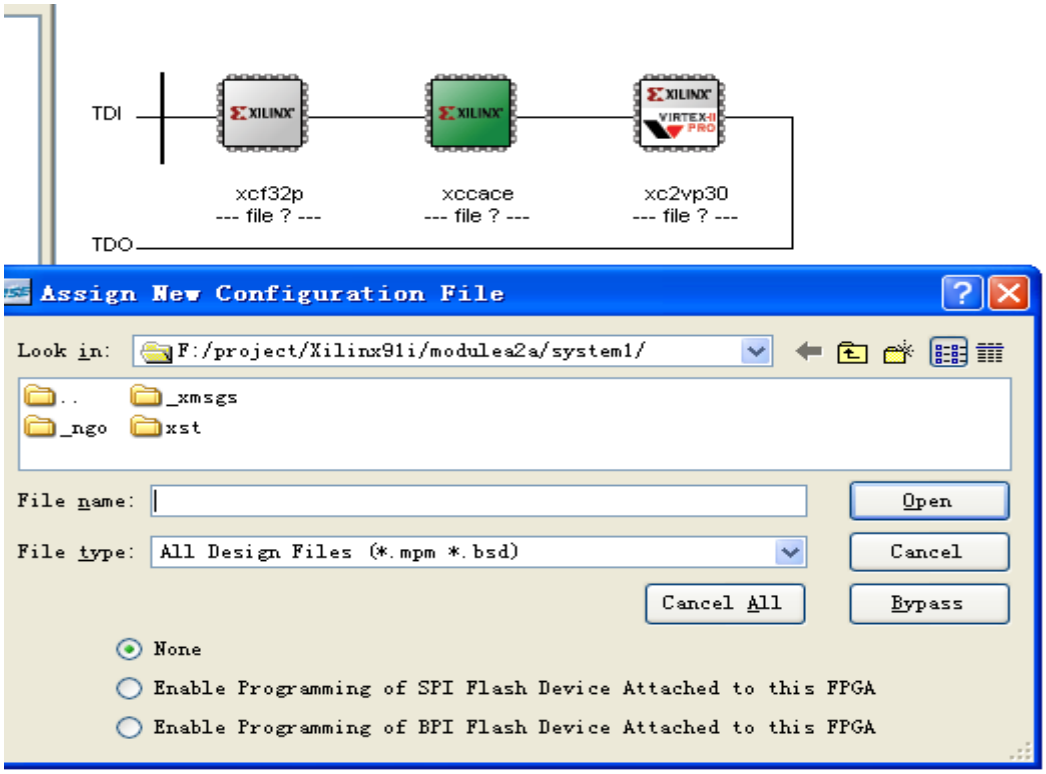
Double click on Configure Device (iMPACT)



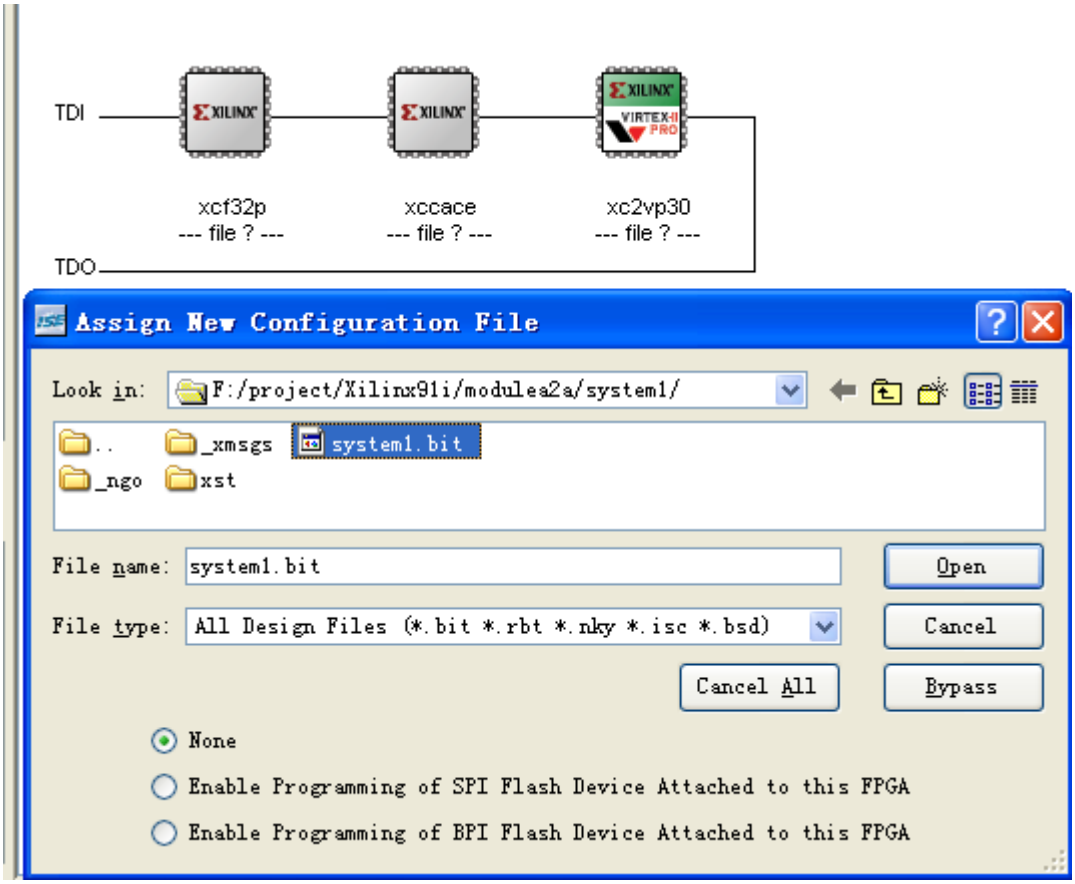
Click on Finish



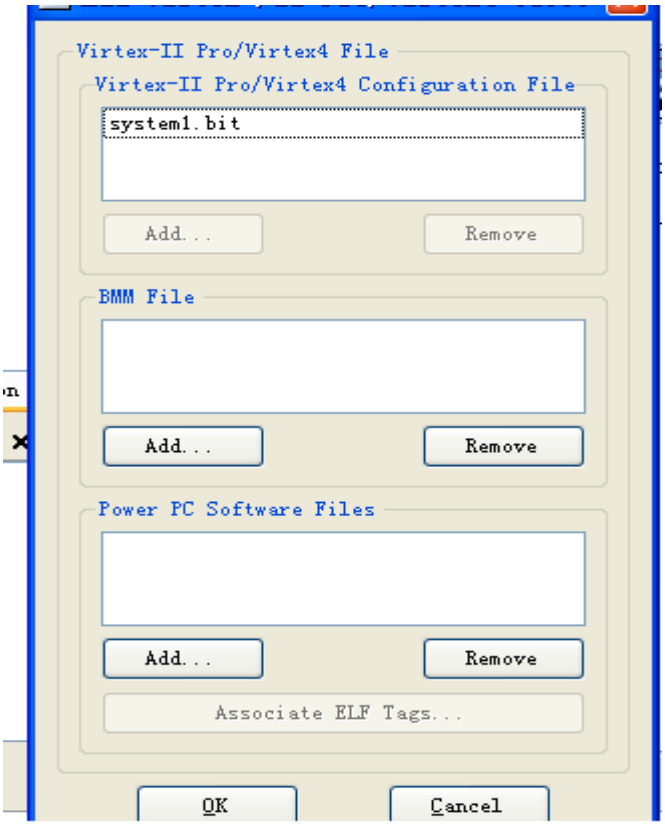
Click on Cancel



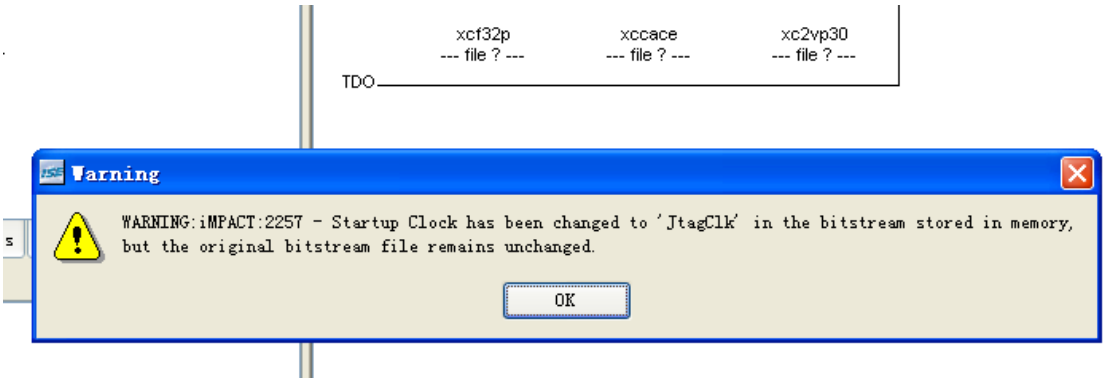
Click on Cancel



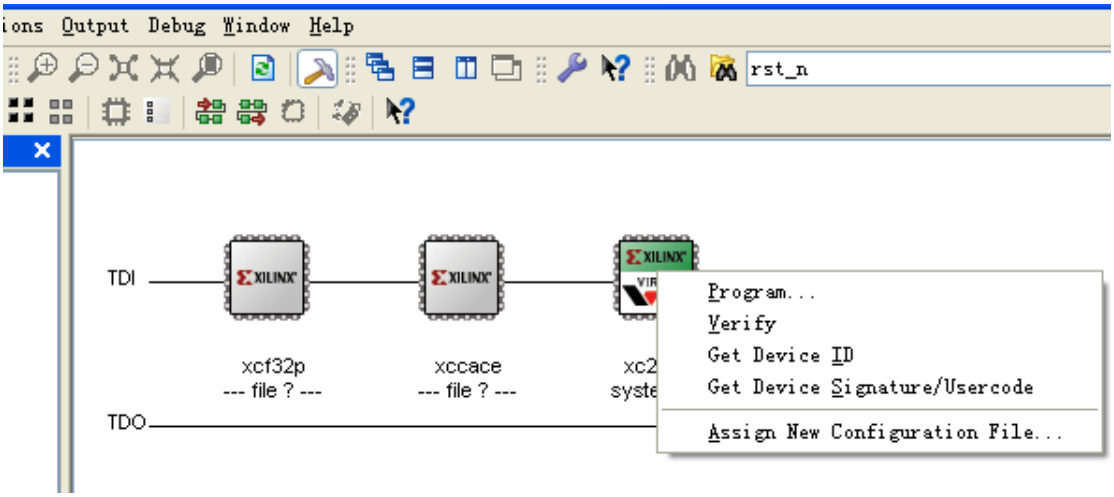
Select System1.bit and click Open



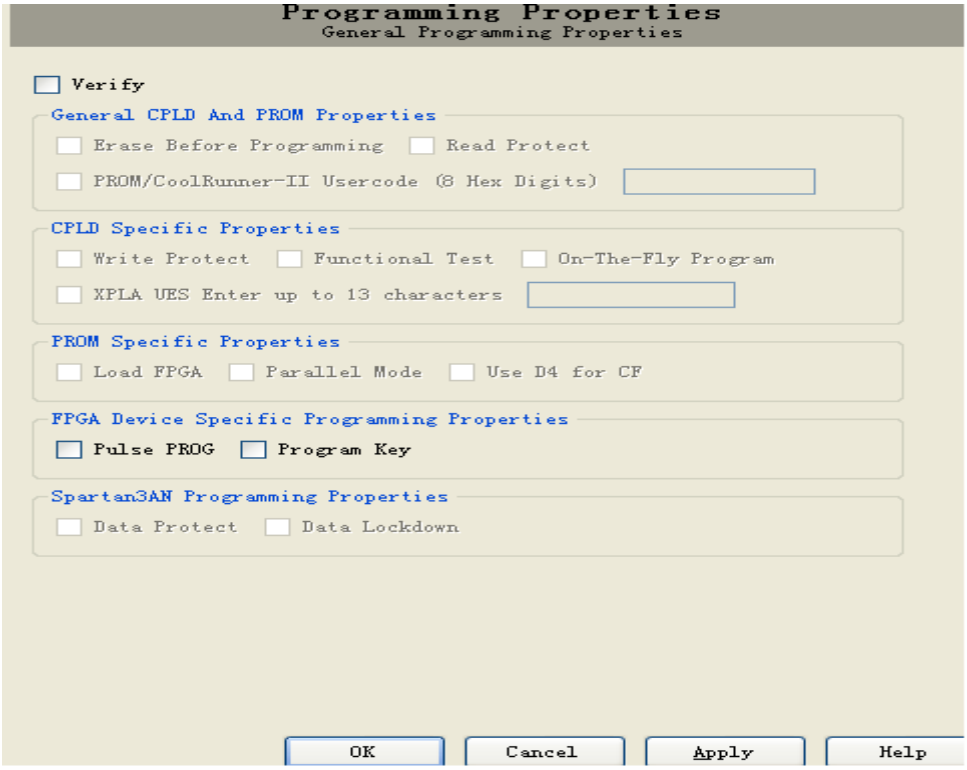
Click on Ok



Ignore the warning and click on ok



Right click on the third device and select Program.



Click on OK

Than you will see the screen connected to the XUP board is red.

congratulations