xios Hogeschool Limburg

Lab 4:

Integrating a picoblaze processor in LabVIEW FPGA by use of CLIP node

Keywords: LabVIEW, LabVIEW FPGA, Xilinx SPARTAN3E Starter Kit, VHDL, picoblaze, assembler, CLIP node.

Introduction

Welcome to Lab 4 in the series of programming a SPARTAN3E Starter Kit by use of LabVIEW FPGA. These labs are created by Vincent Claes. If you encounter problems using this labs or want some advice/consultancy on LabVIEW and especially LabVIEW FPGA you can always contact the author.

These labs are free to use however to show respect to the author please email him when you use them with your contact details (feedback is also welcome).

Contact Information:
Vincent Claes
claesvincent@gmail.com
http://www.linkedin.com/in/vincentclaes

Software Requirements:

- LabVIEW 8.6 or above
- LabVIEW 8.6 FPGA module
- XUP Spartan3E starter board: download for free from: https://lumen.ni.com/nicif/us/infolvfpgaxilsprtn/content.xhtml
- CLIP XML Generator (CXG) 1.1.0 or above (see NI website)
- pBlazIDE (http://www.mediatronix.com/pBlazeIDE.htm)
- KCPSM3.vhd
- ROM_form.coe
- ROM_form.vhd
- KCPSM3.exe

Hardware Requirements:

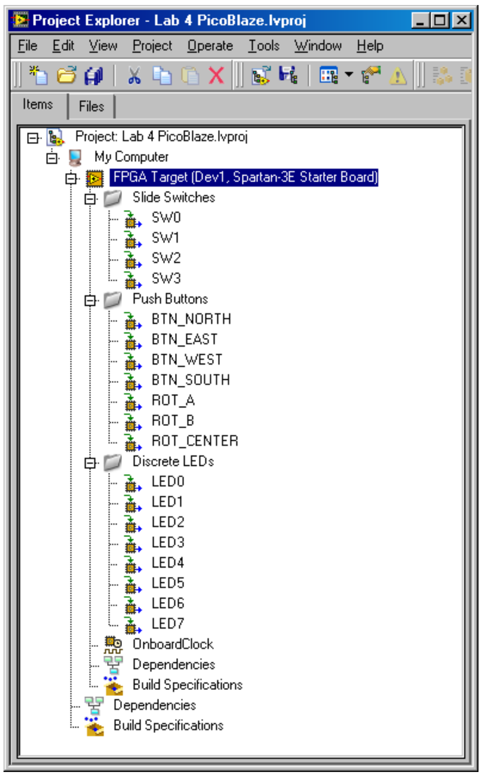
- Xilinx Spartan3E Starter kit: http://www.xilinx.com/products/devkits/HW-SPAR3E-SK-US-G.htm
- User manual: www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf

Knowledge:

- Assembler
- VHDL
- CLIP node
- LabVIEW FPGA

Step 1: Create LabVIEW FPGA Project for Xilinx Spartan 3E starter board.

Like in all the previous labs you have to setup a LabVIEW FPGA Project where you add the Spartan 3E Starter board as a target. If you have troubles doing this you best review labs 1-2 and 3.
Add as FPGA I/O the Slides Switches, the Push Buttons and the Discrete LEDs (see figure below).



Project: Lab 4 PicoBlaze.lvproj
- My Computer
  - FPGA Target (Dev1, Spartan-3E Starter Board)
    - Slide Switches
      - SW0
      - SW1
      - SW2
      - SW3
    - Push Buttons
      - BTN_NORTH
      - BTN_EAST
      - BTN_WEST
      - BTN_SOUTH
      - ROT_A
      - ROT_B
      - ROT_CENTER
    - Discrete LEDs
      - LED0
      - LED1
      - LED2
      - LED3
      - LED4
      - LED5
      - LED6
      - LED7
    - OnboardClock
    - Dependencies
    - Build Specifications
  - Dependencies
  - Build Specifications

Step 2: use pBlazIDE to program a psm file that will run on the picoblaze softcore processor.

Make sure you download pBlazIDE to program your picoblaze application in assembler. Download it from: http://www.mediatronix.com/pBlazeIDE.htm.

For this lab we don't use it because we are going to use an assembler program that I have created already for you.

Create a psm file (lvfpga.psm) in notepad and paste the following code into it:

```
;Programmed by Vincent Claes
;http://pwo.fpga.be

INPUT s0, 00
OUTPUT s0, 01
JUMP 000
```

Since this is not a lab on assembler I am not going to explain the code here. If you need support on coding the picoblaze processor with your assembler code please see the following file: http://www.xilinx.com/support/documentation/ip_documentation/ug129.pdf

Step 3: KCPSM3 tool to generate a VHDL file.

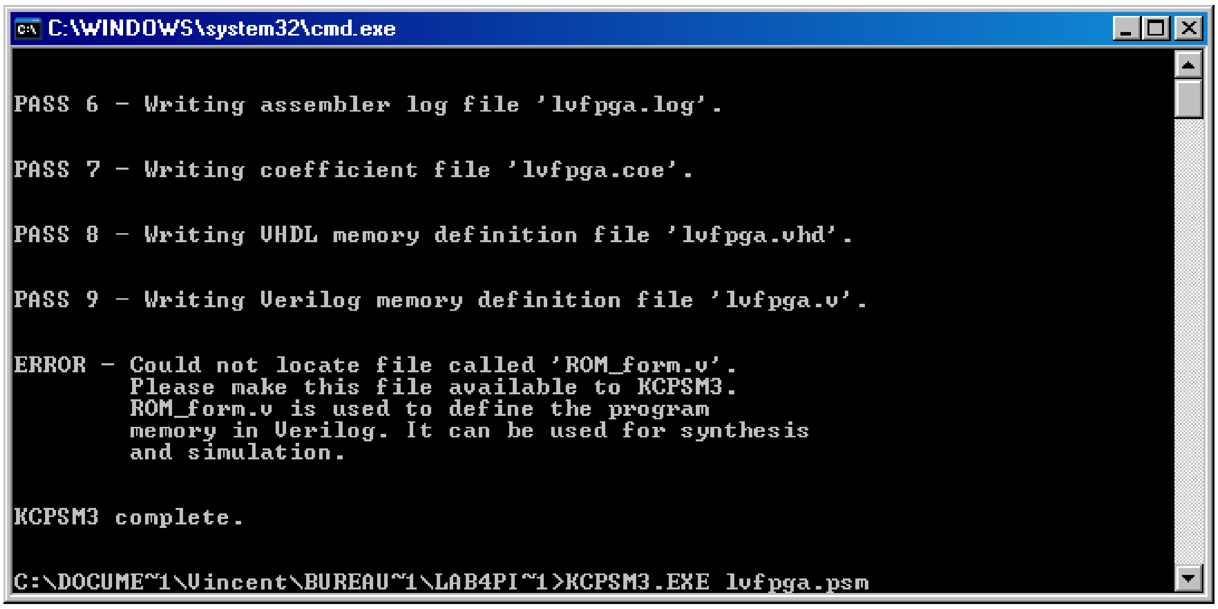
In this step you are going to generate a VHDL file that contains your program (see picoblaze user guide ug129.pdf)

Copy the following files into the map where you have the labviewfpga.psm file created: KCPSM3.EXE, ROM_form.vhd, ROM_form.coe and kcpsm3.vhd . You can find these files in the solution zip file.
Go to the Windows command prompt and execute the following command in the map where you have placed these files:

```
KCPSM3.EXE lvfpga.psm
```

If this exe gives the following output it is ok (the error message about ROM_form.v is normal since we use the

ROM_form.hdl - search in google for difference in verilog and vhdl hardware description languages):



```
C:\WINDOWS\system32\cmd.exe

PASS 6 - Writing assembler log file 'lvfpga.log'.

PASS 7 - Writing coefficient file 'lvfpga.coe'.

PASS 8 - Writing VHDL memory definition file 'lvfpga.vhd'.

PASS 9 - Writing Verilog memory definition file 'lvfpga.v'.

ERROR - Could not locate file called 'ROM_form.v'.
        Please make this file available to KCPSM3.
        ROM_form.v is used to define the program
        memory in Verilog. It can be used for synthesis
        and simulation.

KCPSM3 complete.

C:\DOCUME~1\Vincent\BUREAU~1\LAB4PI~1>KCPSM3.EXE lvfpga.psm
```

Check if in your map the lvfpga.hdl file is created.
Most of the problems I have seen are users who use a different ROM_form.hdl that includes a BSCAN macro. LabVIEW FPGA will then generate an error because you are using 2 BSCAN blocks and there is only 1 available.

Step 4: Build top VHDL file to connect the ROM and KCPSM3 hdl file.

Now we have a description for our program in vhdl (lvfpga.vhd) and our softcore picoblaze processor (kcpsm3.vhd). We need to connect those 2 vhd files together to have a working system. I will be doing this in another vhd file. You can do this also in LabVIEW (maybe in another future lab ;-))

When building a new top vhd file we need to think about LabVIEW variable types. You can read some thinks about it on the NI website: http://zone.ni.com/devzone/cda/tut/p/id/7444.

Create a new .vhd file and name it picoblazesystem.vhd. (You can create vhd files in notepad.)
Paste the following code into it:

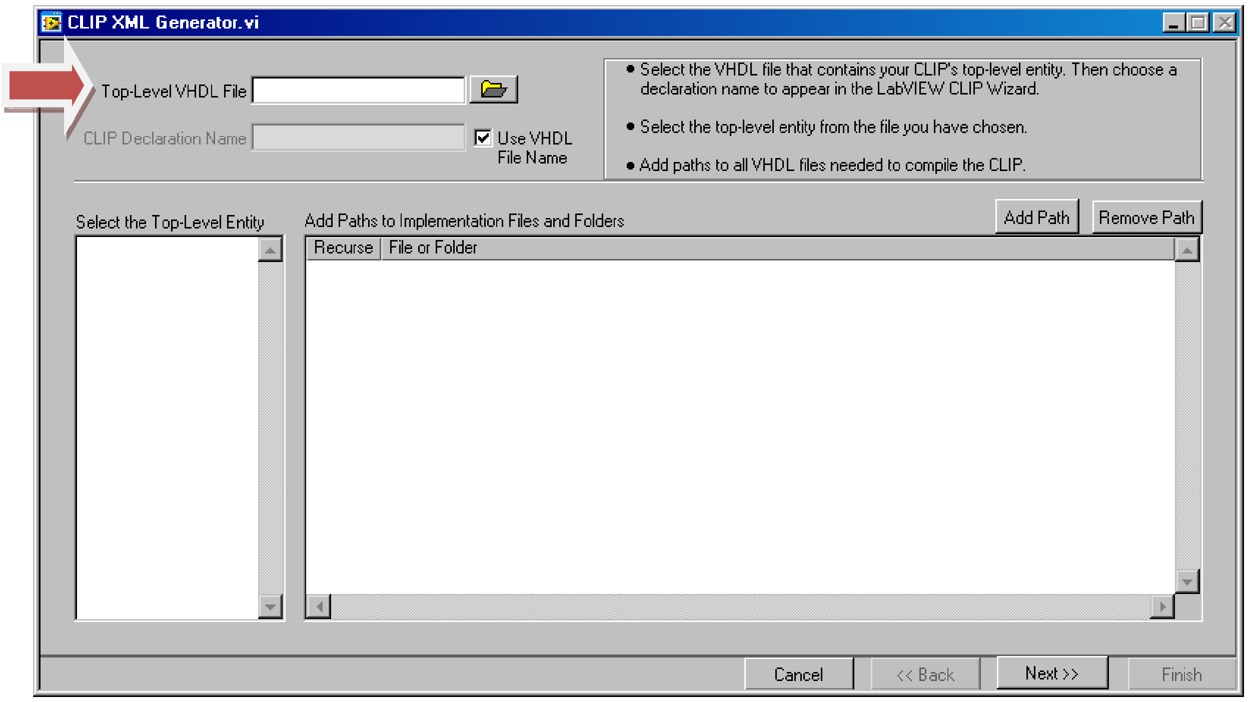
```
-- Created by Vincent Claes
-- http://pwo.fpga.be

library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity picoblazesystem is
    Port (       switches : in std_logic_vector(7 downto 0);
                 LEDS : out std_logic_vector(7 downto 0);
                 clk : in std_logic);
    end picoblazesystem;

architecture Behavioral of picoblazesystem is

-- declaration of KCPSM3 (always use this declaration to call up PicoBlaze core)
  component kcpsm3
    Port (      address : out std_logic_vector(9 downto 0);
            instruction : in std_logic_vector(17 downto 0);
                port_id : out std_logic_vector(7 downto 0);
           write_strobe : out std_logic;
               out_port : out std_logic_vector(7 downto 0);
            read_strobe : out std_logic;
                in_port : in std_logic_vector(7 downto 0);
              interrupt : in std_logic;
          interrupt_ack : out std_logic;
                  reset : in std_logic;
                    clk : in std_logic);
    end component;
-- declaration of program memory (here you will specify the entity name as your .psm
prefix name)
  component lvfpga
    Port (      address : in std_logic_vector(9 downto 0);
            instruction : out std_logic_vector(17 downto 0);
                    clk : in std_logic);
    end component;
-- Signals used to connect PicoBlaze core to program memory and I/O logic
signal address       : std_logic_vector(9 downto 0);
signal instruction   : std_logic_vector(17 downto 0);
signal port_id       : std_logic_vector(7 downto 0);
signal out_port      : std_logic_vector(7 downto 0);
signal in_port       : std_logic_vector(7 downto 0);
signal write_strobe  : std_logic;
signal read_strobe   : std_logic;
signal interrupt_ack : std_logic;
-- the following 2 inputs are assigend inactive values since they are unused in this
example
signal reset         : std_logic :='0';
signal interrupt     : std_logic :='0';
-- Start of circuit description
begin
  -- Instantiating the PicoBlaze core
  processor: kcpsm3
    port map(      address => address,
            instruction => instruction,
                port_id => port_id,
            write_strobe => write_strobe,
               out_port => out_port,
             read_strobe => read_strobe,
                in_port => in_port,
              interrupt => interrupt,
          interrupt_ack => interrupt_ack,
                  reset => reset,
                    clk => clk);
 -- Instantiating the program memory
 program: lvfpga
   port map(      address => address,
            instruction => instruction,
                    clk => clk);
 -- Connect I/O of PicoBlaze
 in_port <= switches;
 LEDS <= out_port;
end Behavioral;
```

Step 5: Use CLIP XML Generator to generate the XML file.

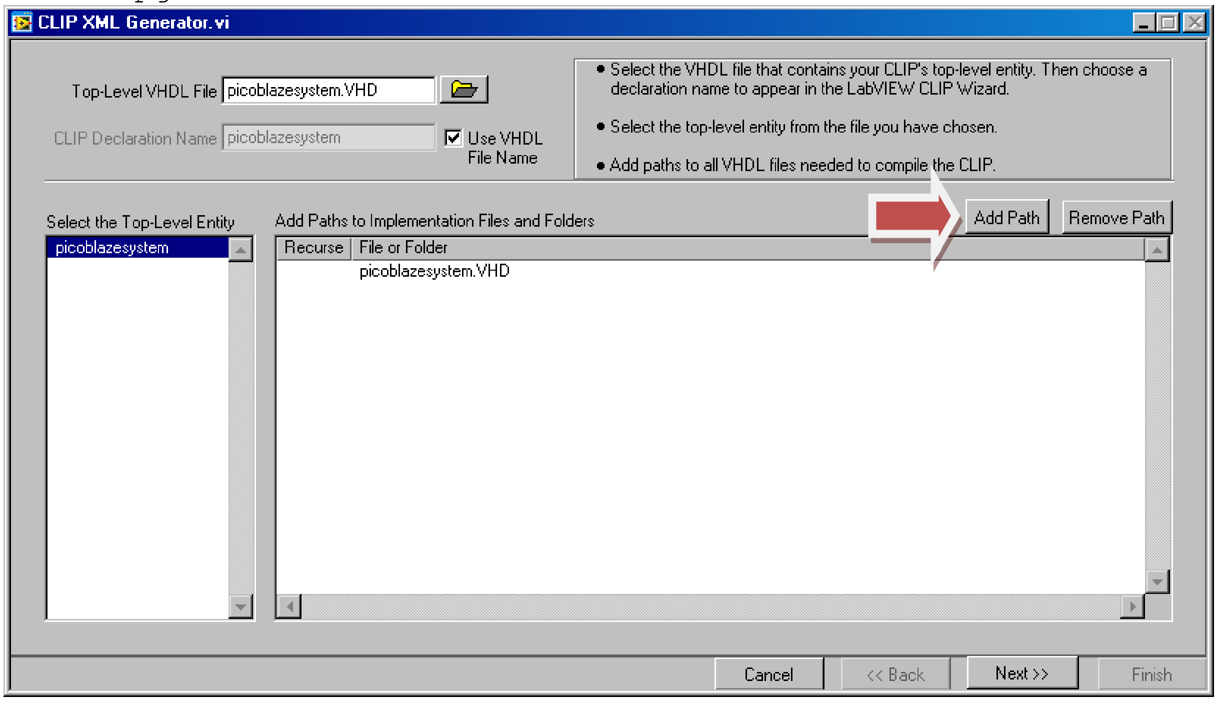
Download the CLIP XML Generator from the NI website: http://zone.ni.com/devzone/cda/epd/p/id/6068

Startup the CLIP XML Generator and select the Top-level VHDL file (picoblazesystem.vhd).



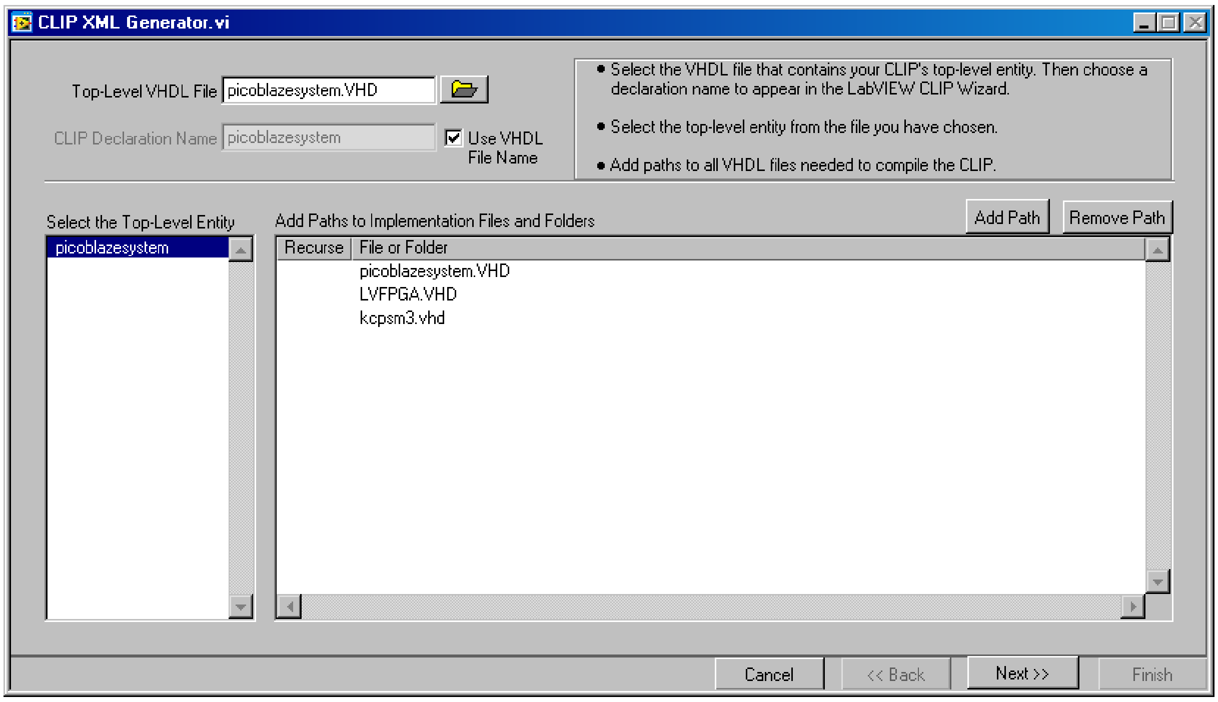
- Select the VHDL file that contains your CLIP's top-level entity. Then choose a declaration name to appear in the LabVIEW CLIP Wizard.
- Select the top-level entity from the file you have chosen.
- Add paths to all VHDL files needed to compile the CLIP.

Click "Add Path" and add the following vhd files: kcpsm3.vhd and lvfpga.vhd



Top-Level VHDL File: picoblazesystem.VHD
CLIP Declaration Name: picoblazesystem
☑ Use VHDL File Name
Select the Top-Level Entity: picoblazesystem
File or Folder: picoblazesystem.VHD

- Select the VHDL file that contains your CLIP's top-level entity. Then choose a declaration name to appear in the LabVIEW CLIP Wizard.
- Select the top-level entity from the file you have chosen.
- Add paths to all VHDL files needed to compile the CLIP.



Top-Level VHDL File: picoblazesystem.VHD

CLIP Declaration Name: picoblazesystem

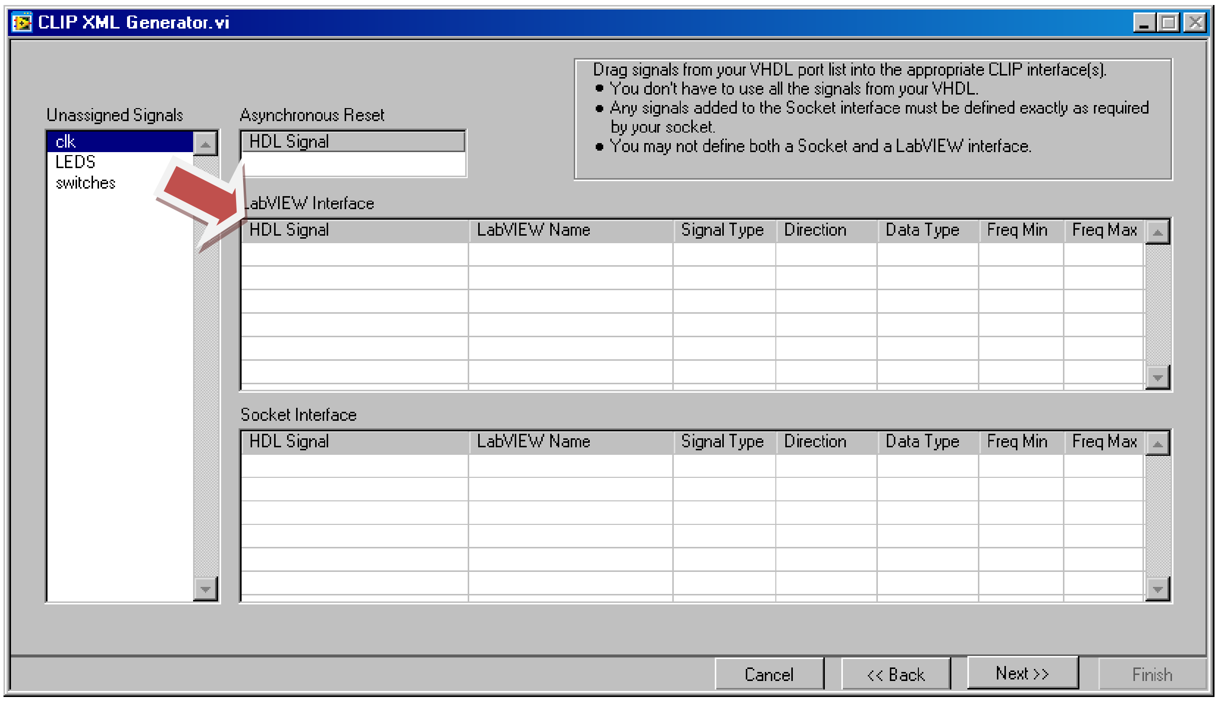
☑ Use VHDL File Name

- Select the VHDL file that contains your CLIP's top-level entity. Then choose a declaration name to appear in the LabVIEW CLIP Wizard.
- Select the top-level entity from the file you have chosen.
- Add paths to all VHDL files needed to compile the CLIP.

Select the Top-Level Entity: picoblazesystem

Recurse	File or Folder
	picoblazesystem.VHD
	LVFPGA.VHD
	kcpsm3.vhd

Click the "next button".



Drag signals from your VHDL port list into the appropriate CLIP interface(s).
- You don't have to use all the signals from your VHDL.
- Any signals added to the Socket interface must be defined exactly as required by your socket.
- You may not define both a Socket and a LabVIEW interface.

Unassigned Signals:
clk
LEDS
switches

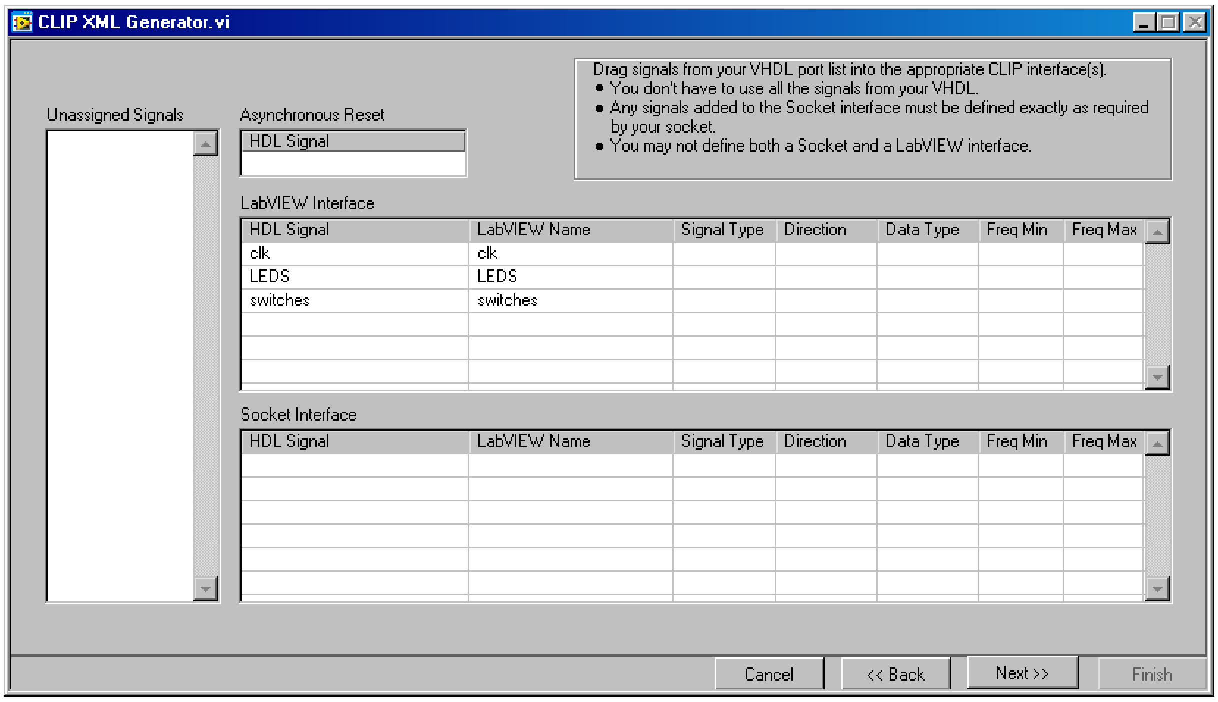
LabVIEW Interface

HDL Signal	LabVIEW Name	Signal Type	Direction	Data Type	Freq Min	Freq Max

Socket Interface

HDL Signal	LabVIEW Name	Signal Type	Direction	Data Type	Freq Min	Freq Max

Move the "clk", "LEDS" and "switches" to the LabVIEW Interface "HDL Signal".



CLIP XML Generator.vi

Drag signals from your VHDL port list into the appropriate CLIP interface(s).
- You don't have to use all the signals from your VHDL.
- Any signals added to the Socket interface must be defined exactly as required by your socket.
- You may not define both a Socket and a LabVIEW interface.

Unassigned Signals

Asynchronous Reset

HDL Signal

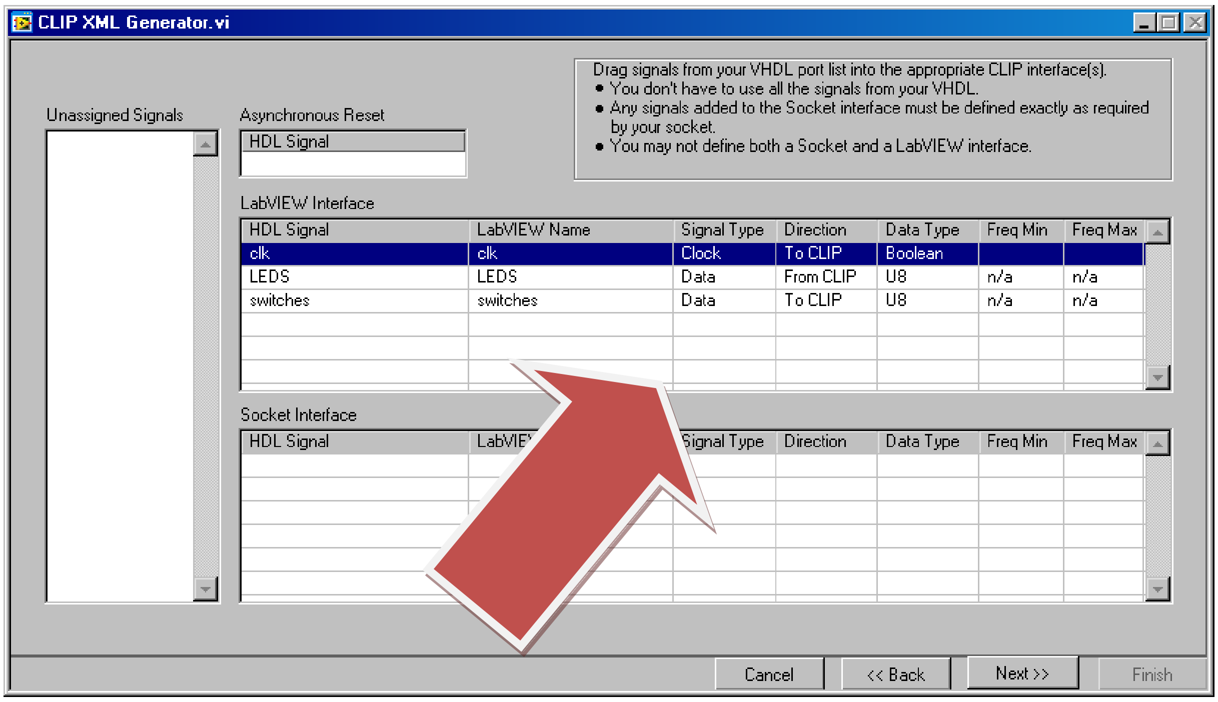
LabVIEW Interface

HDL Signal	LabVIEW Name	Signal Type	Direction	Data Type	Freq Min	Freq Max
clk	clk					
LEDS	LEDS					
switches	switches					

Socket Interface

HDL Signal	LabVIEW Name	Signal Type	Direction	Data Type	Freq Min	Freq Max

Make the following settings for the "Signal Type", "Direction" and "Data Type", the "Freq Min" and "Freq Max" settings for the clock are a little buggy ☺ we will manipulate them in the next paragraph (be carefully, those settings need to be correct!):



CLIP XML Generator.vi

Drag signals from your VHDL port list into the appropriate CLIP interface(s).
- You don't have to use all the signals from your VHDL.
- Any signals added to the Socket interface must be defined exactly as required by your socket.
- You may not define both a Socket and a LabVIEW interface.

Unassigned Signals

Asynchronous Reset

HDL Signal

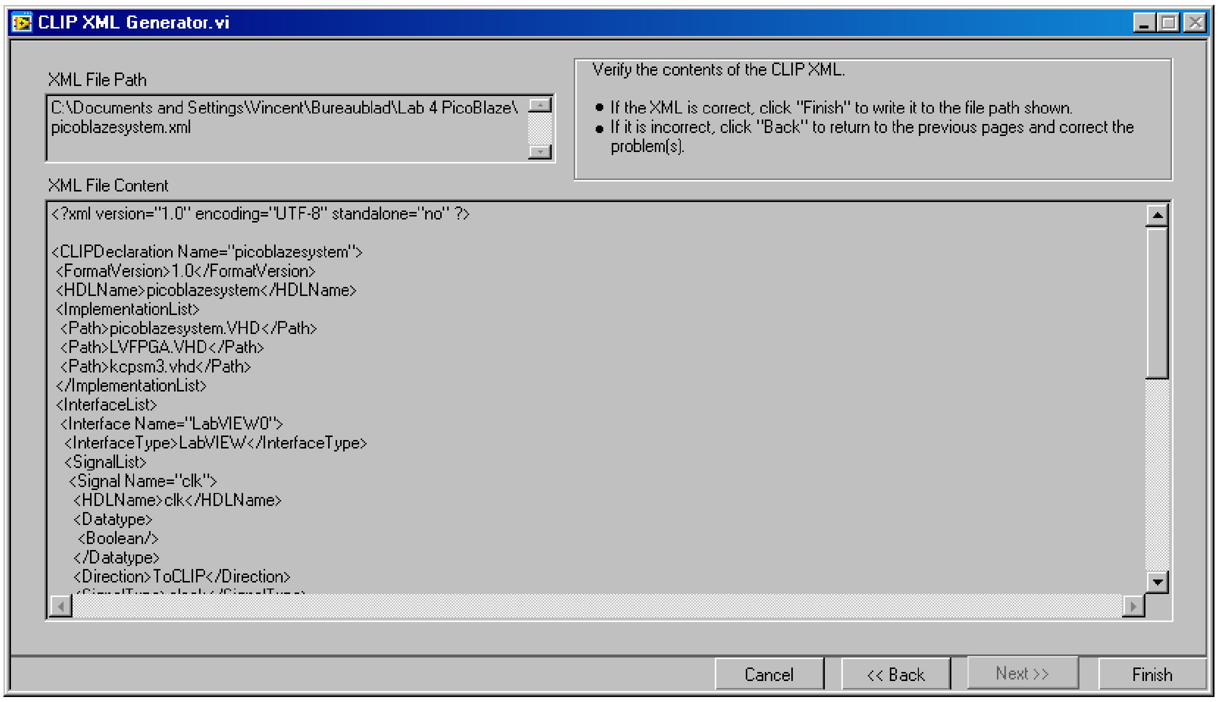
LabVIEW Interface

HDL Signal	LabVIEW Name	Signal Type	Direction	Data Type	Freq Min	Freq Max
clk	clk	Clock	To CLIP	Boolean		
LEDS	LEDS	Data	From CLIP	U8	n/a	n/a
switches	switches	Data	To CLIP	U8	n/a	n/a

Socket Interface

HDL Signal	LabVIE	Signal Type	Direction	Data Type	Freq Min	Freq Max

Click the "Next button".


CLIP XML Generator.vi

XML File Path

C:\Documents and Settings\Vincent\Bureaublad\Lab 4 PicoBlaze\picoblazesystem.xml

Verify the contents of the CLIP XML.

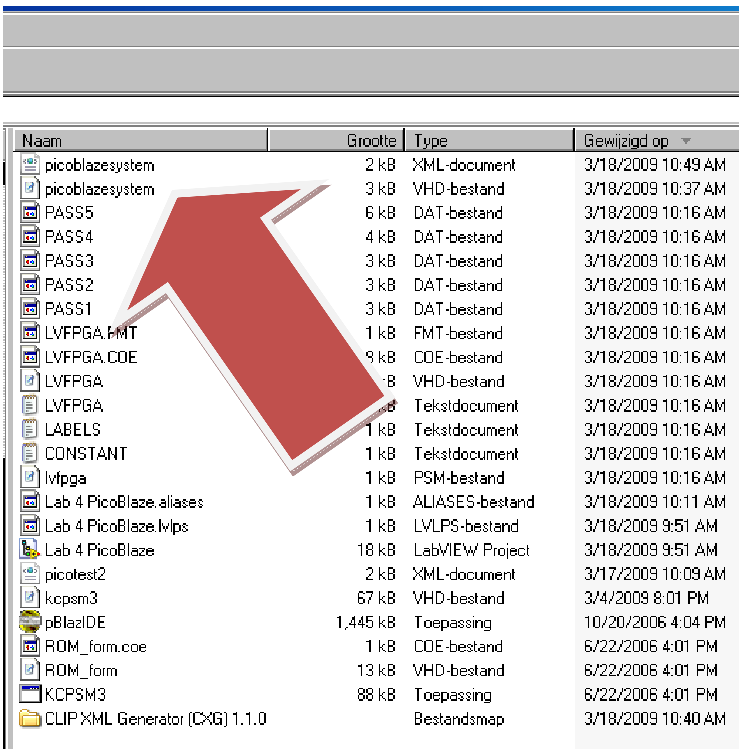
- If the XML is correct, click "Finish" to write it to the file path shown.
- If it is incorrect, click "Back" to return to the previous pages and correct the problem(s).

XML File Content

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<CLIPDeclaration Name="picoblazesystem">
 <FormatVersion>1.0</FormatVersion>
 <HDLName>picoblazesystem</HDLName>
 <ImplementationList>
  <Path>picoblazesystem.VHD</Path>
  <Path>LVFPGA.VHD</Path>
  <Path>kcpsm3.vhd</Path>
 </ImplementationList>
 <InterfaceList>
  <Interface Name="LabVIEW0">
   <InterfaceType>LabVIEW</InterfaceType>
   <SignalList>
    <Signal Name="clk">
     <HDLName>clk</HDLName>
     <Datatype>
      <Boolean/>
     </Datatype>
     <Direction>ToCLIP</Direction>
```

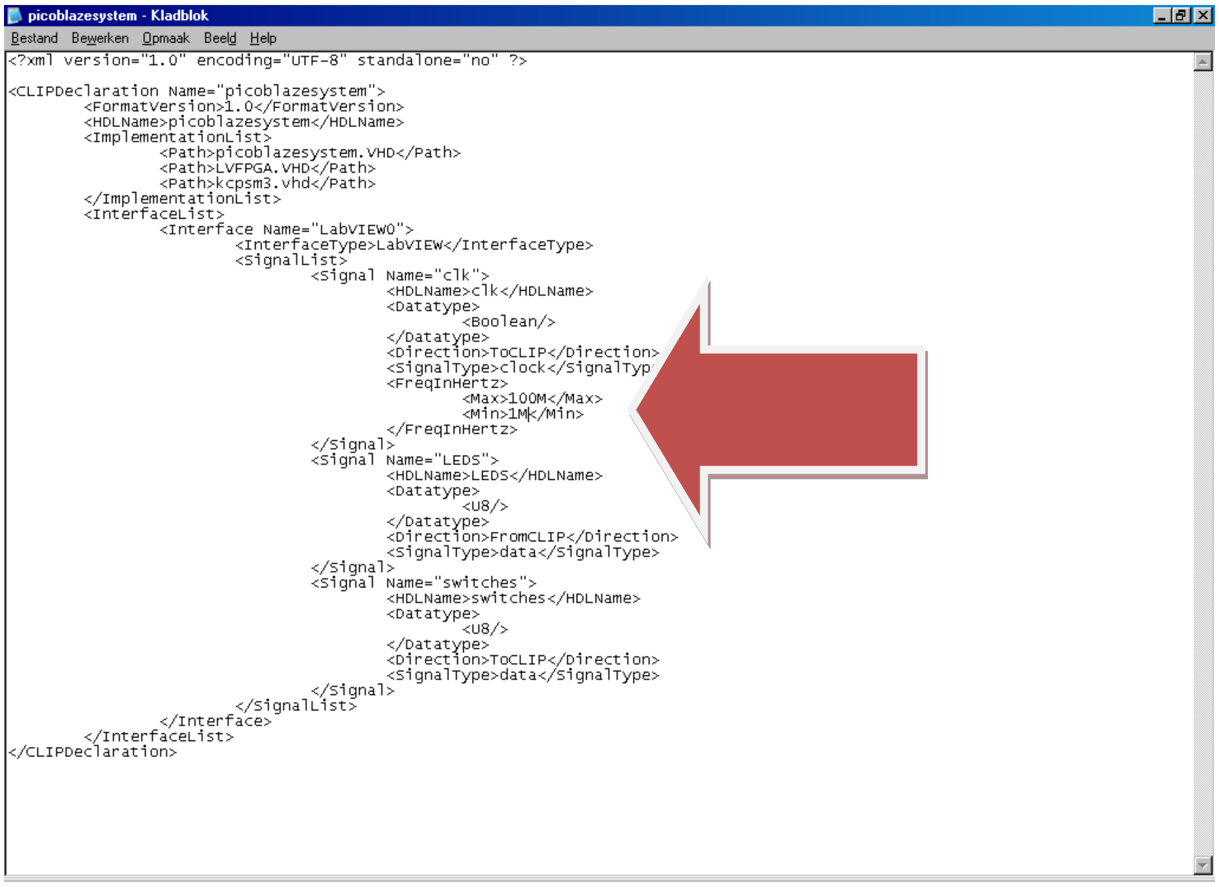
Click the "Finish" button.

Now go back to your map where you where working and see if there is an picblazesystem.xml file. Open this fill with notepad. In notepad we are going to set the "Freq Min" and "Freq Max" settings for the clock.



Naam	Grootte	Type	Gewijzigd op
picoblazesystem	2 kB	XML-document	3/18/2009 10:49 AM
picoblazesystem	3 kB	VHD-bestand	3/18/2009 10:37 AM
PASS5	6 kB	DAT-bestand	3/18/2009 10:16 AM
PASS4	4 kB	DAT-bestand	3/18/2009 10:16 AM
PASS3	3 kB	DAT-bestand	3/18/2009 10:16 AM
PASS2	3 kB	DAT-bestand	3/18/2009 10:16 AM
PASS1	3 kB	DAT-bestand	3/18/2009 10:16 AM
LVFPGA.FMT	1 kB	FMT-bestand	3/18/2009 10:16 AM
LVFPGA.COE	9 kB	COE-bestand	3/18/2009 10:16 AM
LVFPGA	kB	VHD-bestand	3/18/2009 10:16 AM
LVFPGA	kB	Tekstdocument	3/18/2009 10:16 AM
LABELS	1 kB	Tekstdocument	3/18/2009 10:16 AM
CONSTANT	1 kB	Tekstdocument	3/18/2009 10:16 AM
lvfpga	1 kB	PSM-bestand	3/18/2009 10:16 AM
Lab 4 PicoBlaze.aliases	1 kB	ALIASES-bestand	3/18/2009 10:11 AM
Lab 4 PicoBlaze.lvlps	1 kB	LVLPS-bestand	3/18/2009 9:51 AM
Lab 4 PicoBlaze	18 kB	LabVIEW Project	3/18/2009 9:51 AM
picotest2	2 kB	XML-document	3/17/2009 10:09 AM
kcpsm3	67 kB	VHD-bestand	3/4/2009 8:01 PM
pBlazIDE	1,445 kB	Toepassing	10/20/2006 4:04 PM
ROM_form.coe	1 kB	COE-bestand	6/22/2006 4:01 PM
ROM_form	13 kB	VHD-bestand	6/22/2006 4:01 PM
KCPSM3	88 kB	Toepassing	6/22/2006 4:01 PM
CLIP XML Generator (CXG) 1.1.0		Bestandsmap	3/18/2009 10:40 AM

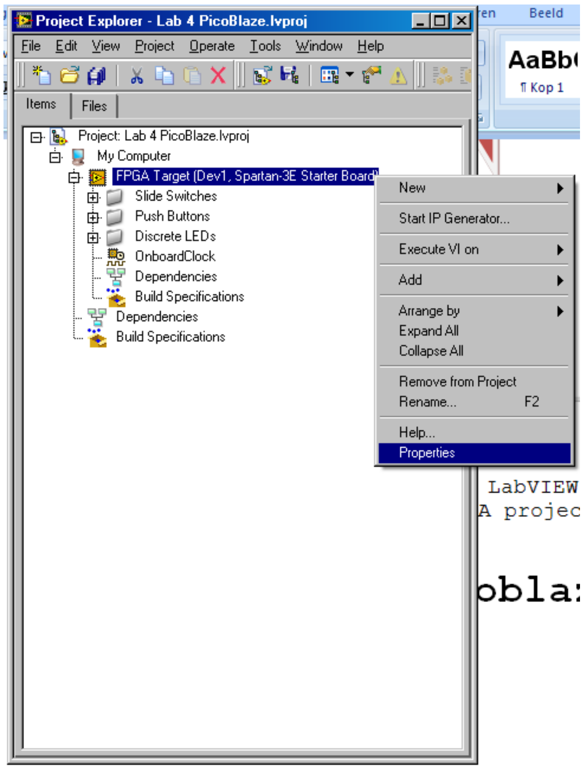
Place the value of 100M between the <Max> </Max> XML tags and the value 1M between the <Min> </Min> tags, save the file and close it.


picoblazesystem - Kladblok
Bestand Bewerken Opmaak Beeld Help

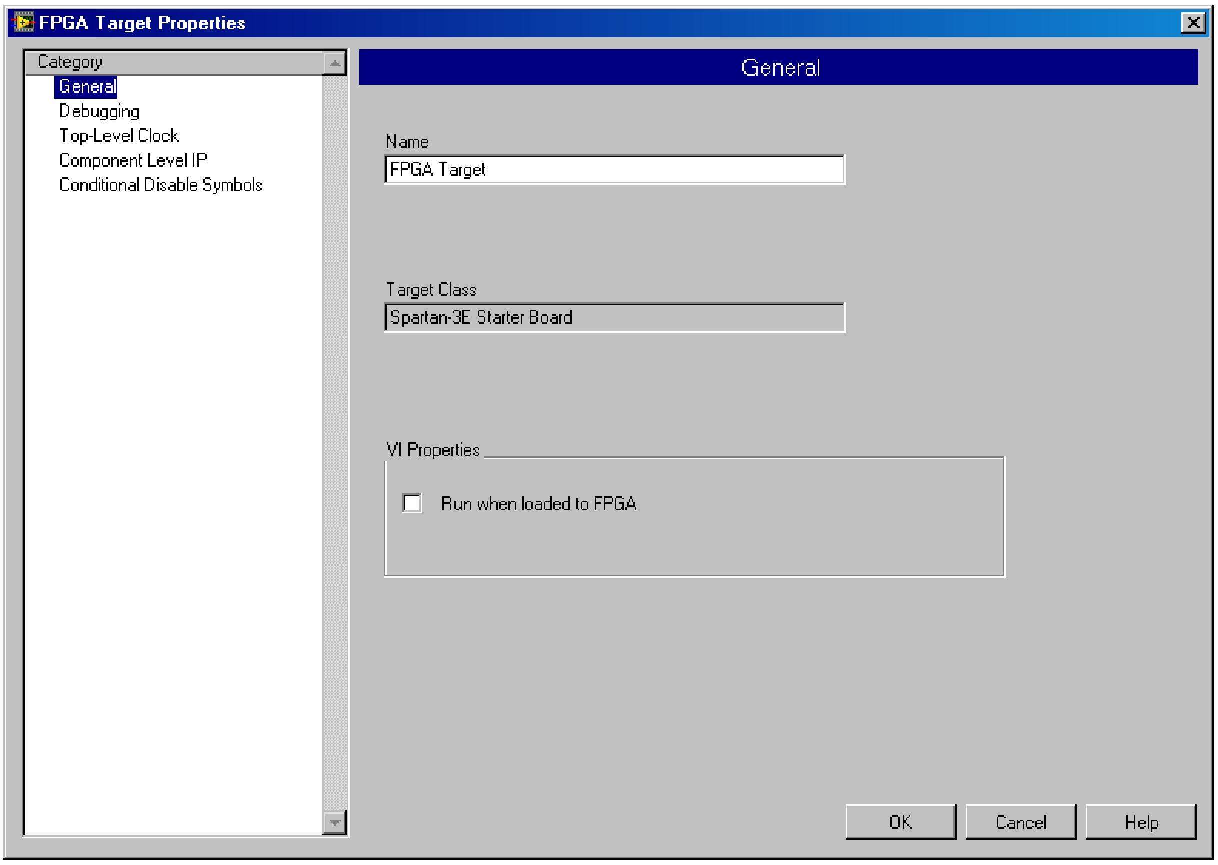
```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<CLIPDeclaration Name="picoblazesystem">
        <FormatVersion>1.0</FormatVersion>
        <HDLName>picoblazesystem</HDLName>
        <ImplementationList>
                <Path>picoblazesystem.VHD</Path>
                <Path>LVFPGA.VHD</Path>
                <Path>kcpsm3.vhd</Path>
        </ImplementationList>
        <InterfaceList>
                <Interface Name="LabVIEW0">
                        <InterfaceType>LabVIEW</InterfaceType>
                        <SignalList>
                                <Signal Name="clk">
                                        <HDLName>clk</HDLName>
                                        <Datatype>
                                                <Boolean/>
                                        </Datatype>
                                        <Direction>ToCLIP</Direction>
                                        <SignalType>clock</SignalTyp
                                        <FreqInHertz>
                                                <Max>100M</Max>
                                                <Min>1M</Min>
                                        </FreqInHertz>
                                </Signal>
                                <Signal Name="LEDS">
                                        <HDLName>LEDS</HDLName>
                                        <Datatype>
                                                <U8/>
                                        </Datatype>
                                        <Direction>FromCLIP</Direction>
                                        <SignalType>data</SignalType>
                                </Signal>
                                <Signal Name="switches">
                                        <HDLName>switches</HDLName>
                                        <Datatype>
                                                <U8/>
                                        </Datatype>
                                        <Direction>ToCLIP</Direction>
                                        <SignalType>data</SignalType>
                                </Signal>
                        </SignalList>
                </Interface>
        </InterfaceList>
</CLIPDeclaration>
```

Step 6: Import CLIP.

Now it is time to go back to your LabVIEW environment. Go to the previously created LabVIEW FPGA project. Do a right mouseclick on your Xilinx Spartan 3E Target.



Check the "run when loaded to FPGA" checkbox and then select the "Component Level IP" selection on the left side of this window.



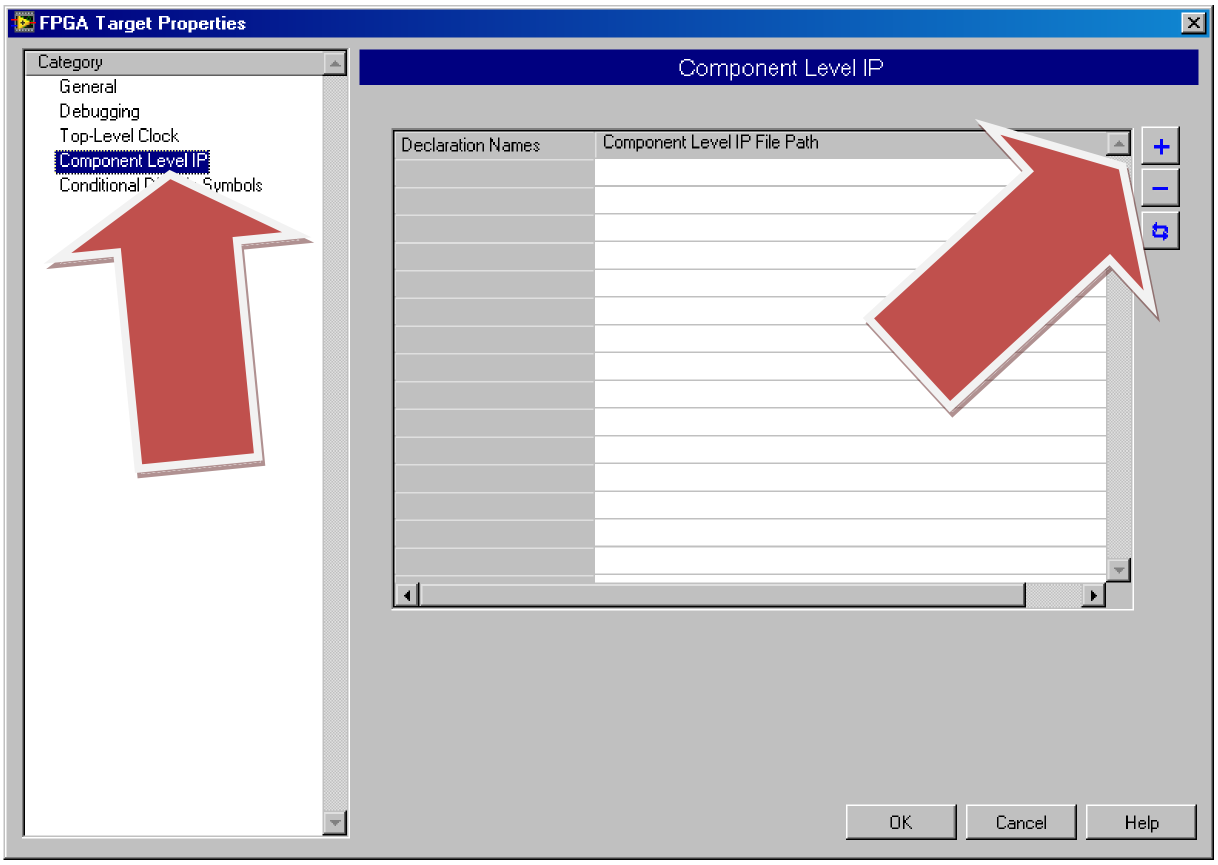
Name: FPGA Target

Target Class: Spartan-3E Starter Board

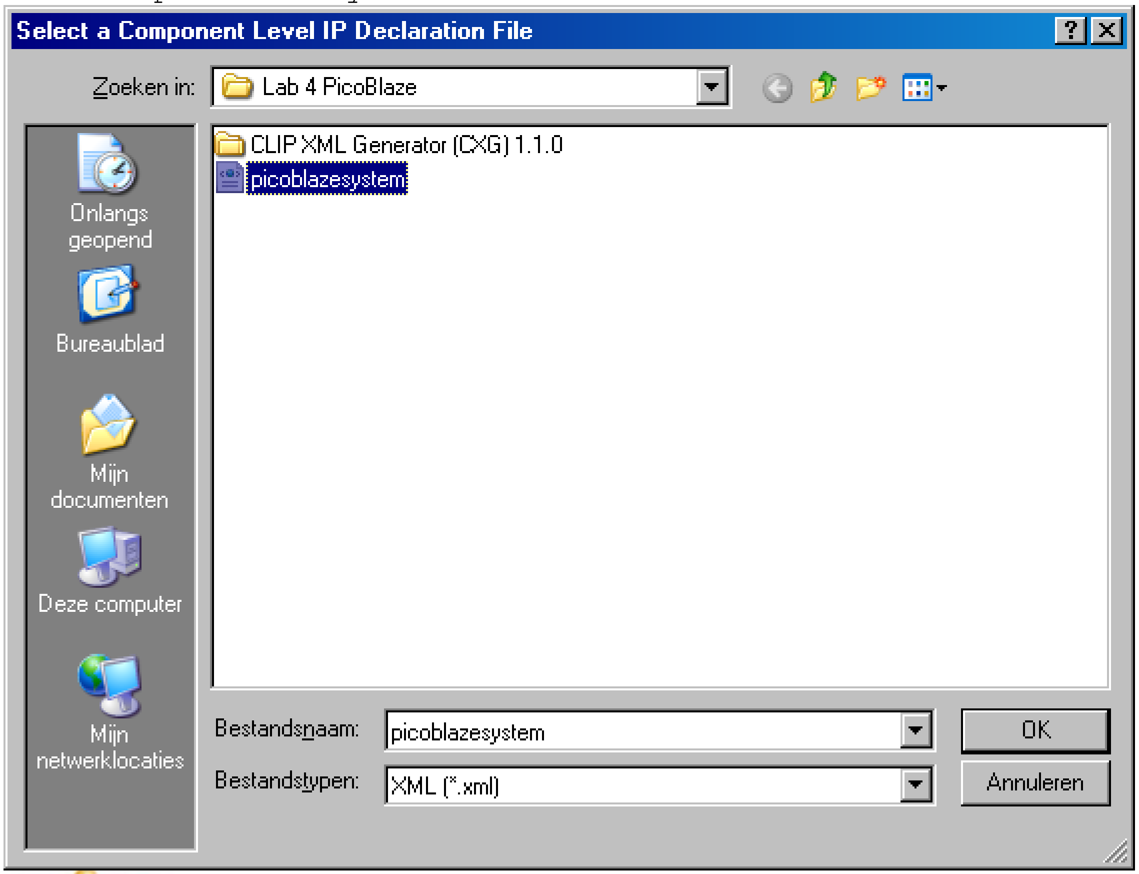
VI Properties

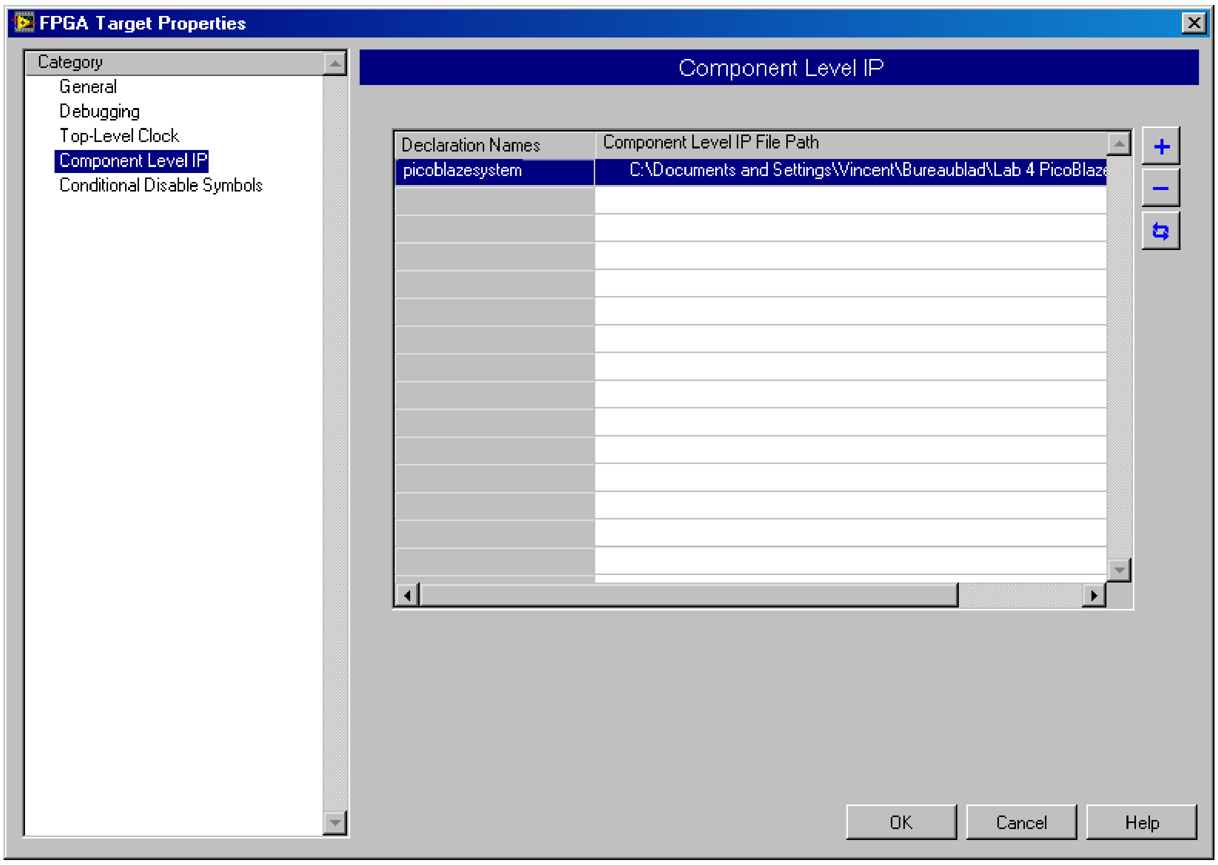
☐ Run when loaded to FPGA

In this new window click the “+” button.



Select “picoblazesystem.xml” and click the “OK button”.

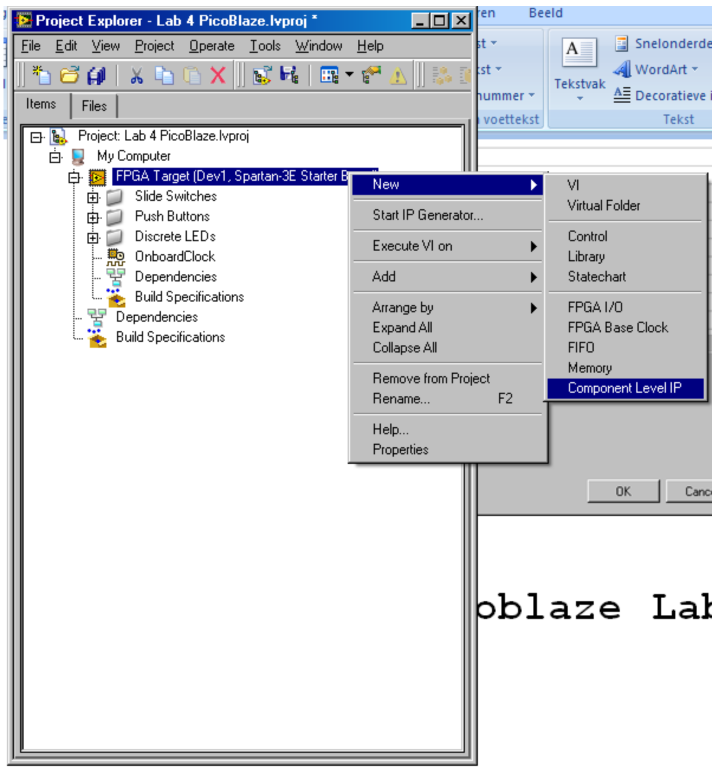




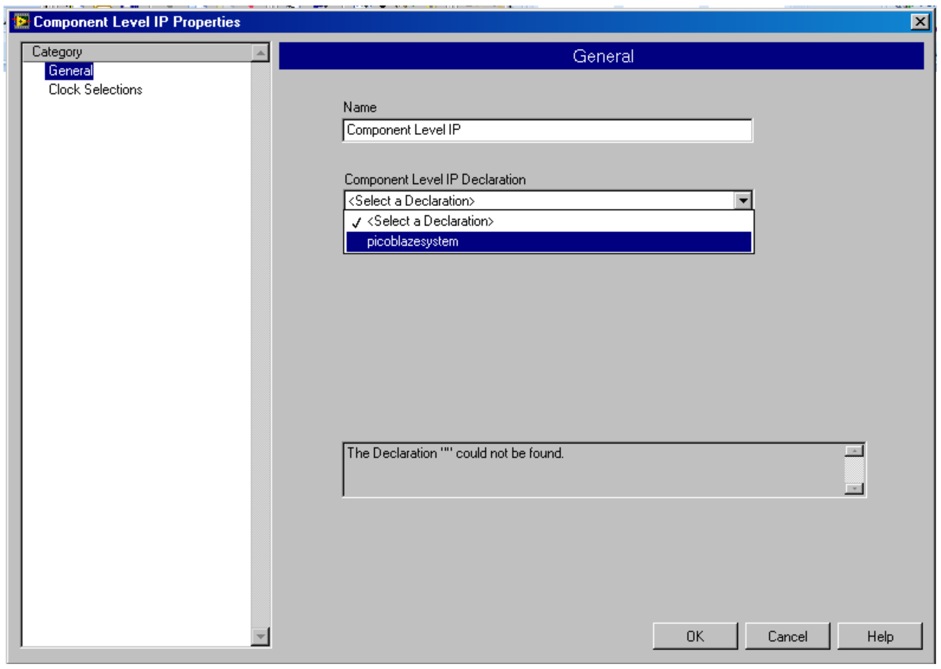
Declaration Names	Component Level IP File Path
picoblazesystem	C:\Documents and Settings\Vincent\Bureaublad\Lab 4 PicoBlaze

Now click the "OK button".

Go back to the project explorer do a right mouse click on the FPGA target and select "New" > "Component Level IP".

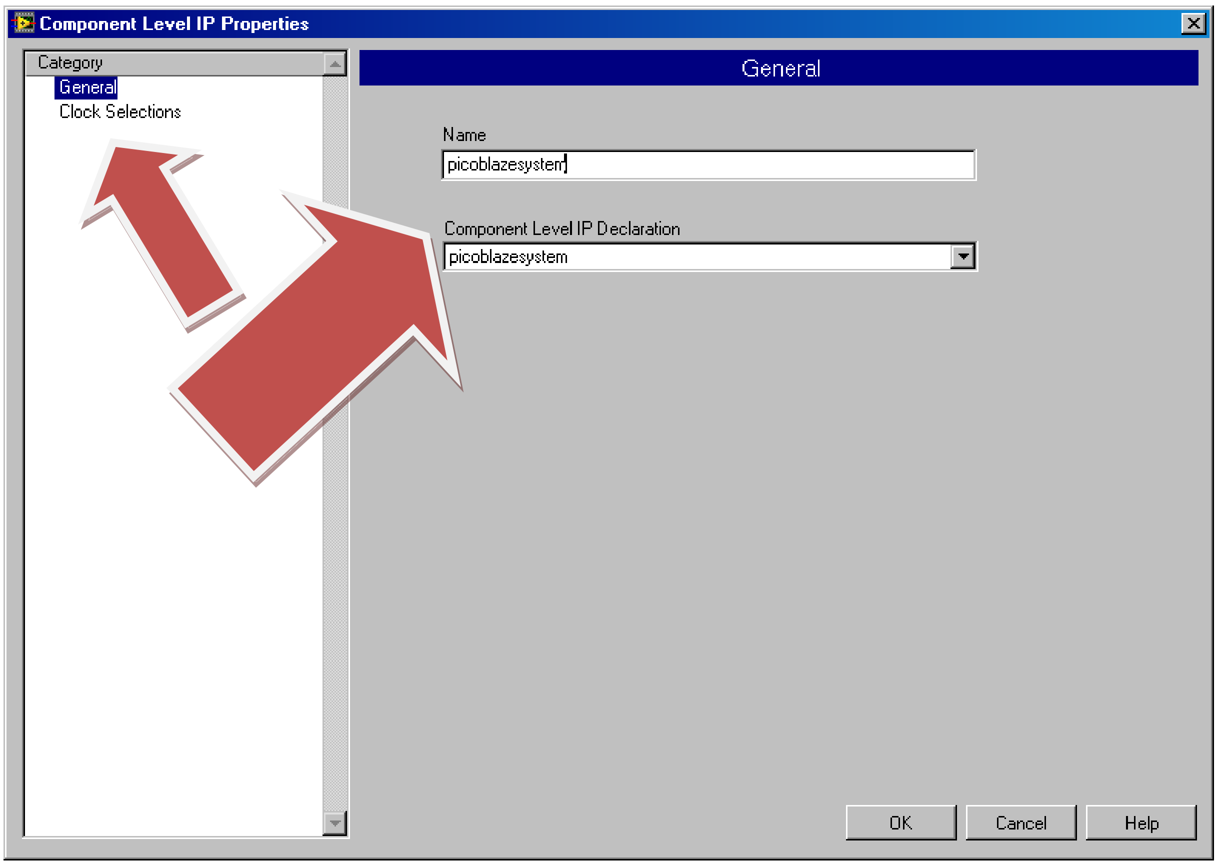


A new window appears; you have to select your "picoblazesystem" in the "Component Level IP Declaration" dropdown box.



The Declaration "" could not be found.

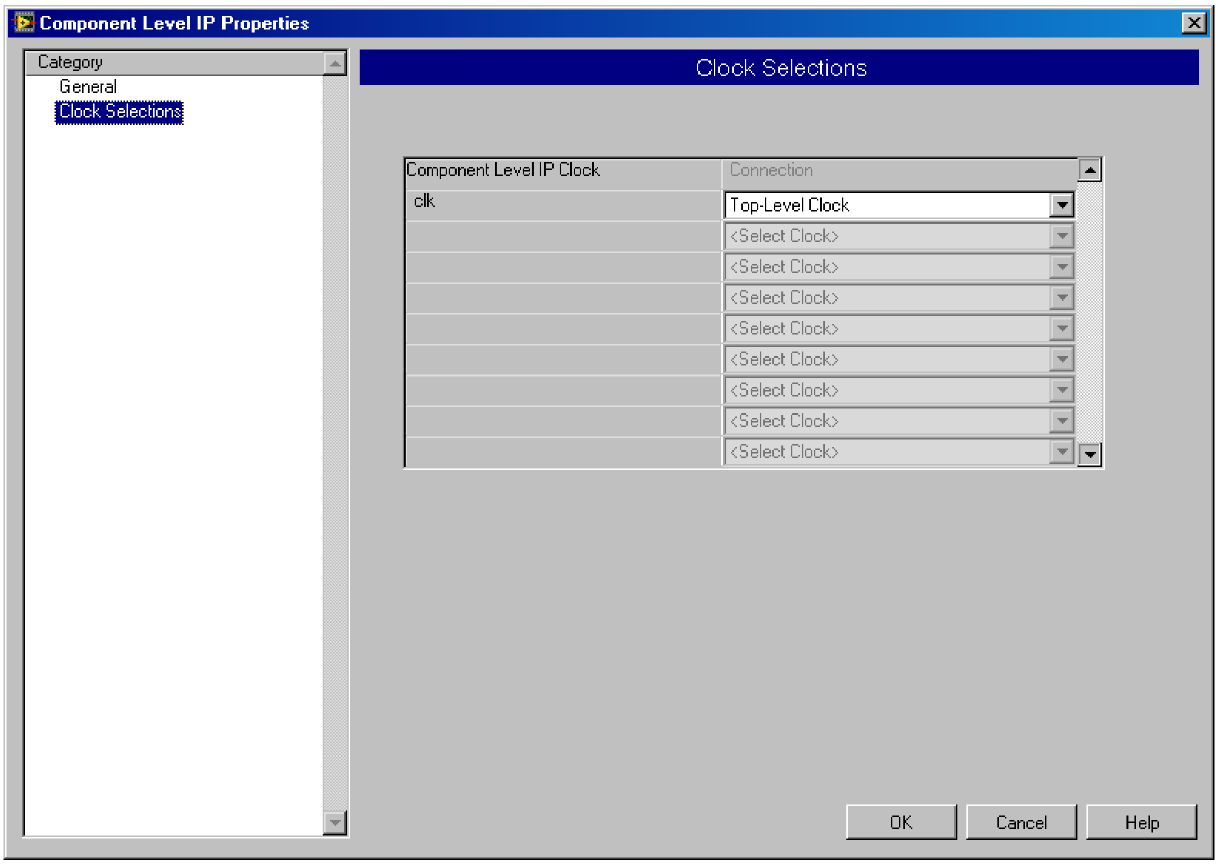
Change also the name to "picoblazesystem". The select in the left corner of the window "Clock Selections".



Name: picoblazesystem

Component Level IP Declaration: picoblazesystem

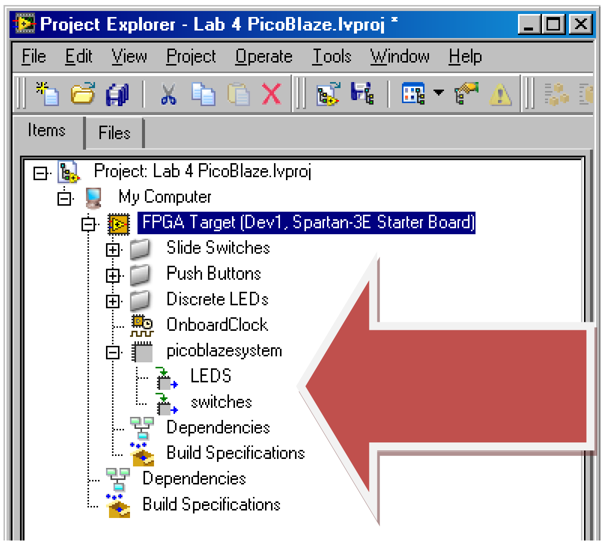
Be sure that for the Component Level IP Clock "clk" the "Top-Level Clock" is selected. Then Click the "OK button".



Component Level IP Properties

Component Level IP Clock	Connection
clk	Top-Level Clock
	<Select Clock>
	<Select Clock>
	<Select Clock>
	<Select Clock>
	<Select Clock>
	<Select Clock>
	<Select Clock>
	<Select Clock>

In LabVIEW Explorer you now have to see the "picoblazesystem" CLIP which has an output port "LEDS" and an input port "switches".



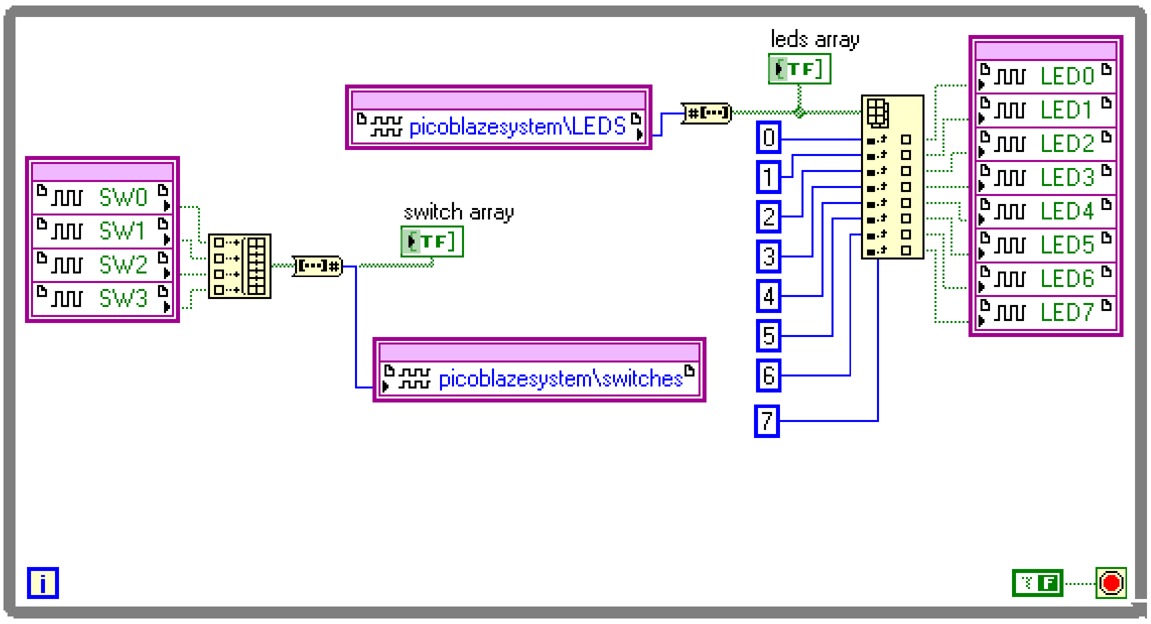
Project Explorer - Lab 4 PicoBlaze.lvproj *

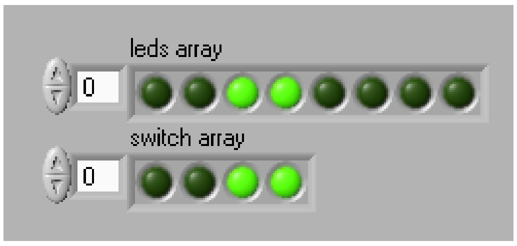
```
Project: Lab 4 PicoBlaze.lvproj
  My Computer
    FPGA Target (Dev1, Spartan-3E Starter Board)
      Slide Switches
      Push Buttons
      Discrete LEDs
      OnboardClock
      picoblazesystem
        LEDS
        switches
      Dependencies
      Build Specifications
    Dependencies
    Build Specifications
```

.

Step 7: Build picoblaze LabVIEW FPGA VI.

Now create a new FPGA vi. You can use the input and output of your CLIP by selecting them in LabVIEW Explorer and dropping them down in your “Block diagram” of the FPGA vi. As an example you could build the following vi:





Now press the “Run” button to have a softcore “picoblaze” running on your Xilinx Spartan 3E starter board with LabVIEW FPGA.

Enjoy.

Vincent Claes
XIOS Hogeschool Limburg
Department of Industrial Sciences and Technology
Universitaire Campus - Agoralaan – Gebouw H
B-3590 Diepenbeek
Belgium
vincent.claes@xios.be
tel.: +32 11 26 00 39
fax: +32 11 26 00 54
mobile: +32 478 35 38 49