Programming **FPGAs**
using **LabVIEW**

“How to program **FPGAs**
without **VHDL** knowledge”

Dr. Vlassis Petousis
Ruđer Bošković Institut (RBI)
Outline

1. Implementing a Boolean function in LabVIEW FPGA on the Xilinx SPARTAN Boards (LAB 01).

2. Embedding VHDL code in a Xilinx SPARTAN FPGA (LAB 02).
F = (A+B)CD

Implementing Logic on FPGA
We have to create an Empty Project
“Right click” on “My Computer” in the Project Explorer view. We select “New” and then “Targets and Devices”.
LabVIEW – FPGA Target

“New target or device”
For this we have to “right click” the FPGA Target
If we go back to the “Project Explorer” view we see the FPGA I/O that we have added. With this I/O we will implement an Exclusive-OR function.
1. The next step is the creation of a hardware program that runs on the Xilinx Spartan hardware target.
2. For this we do again a “right mouse click” on the FPGA target in the “Project Explorer” view. Select “New” VI
We created:

LabVIEW – FPGA VI

5/11/2011
We place 2 I/O Nodes on the Block Diagram of the LabVIEW FPGA VI
To add another FPGA I/O. For this do a right click on the FPGA I/O you just filled with SW0. Select “Add Element”.

LabVIEW – FPGA add Nodes
We can do the same to add another FPGA I/O.
Now we will as an example implement an exclusive-OR Boolean function into the FPGA. From the “Functions Palette” select “Programming” “Boolean” and look for the “Exclusive OR” function. Place this one on the Block diagram.

**XOR Truth Table**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Wire the “SW1” and “SW0” FPGA I/O to the Exclusive-OR inputs.

Wire the output of the function to “LED0”.

5/11/2011
• When we implement the function like presented above the function will run only once. We would like to implement it that it runs continuously.

• For this we place a “While Loop” around it and we wire a Boolean “False constant” to the “stop condition” of this loop.

• The “While Loop” you find on the “Functions Palette”.

5/11/2011
LabVIEW – FPGA “While Loop”
Now it is time to save our VI that we created for the FPGA.
We do this by selecting “File” “Save As...” in either the “Front Panel” or the “Block Diagram”.

![LabVIEW screenshot showing file save options]

![LabVIEW screenshot showing file save dialog box]

5/11/2011
For starting the executing of this VI we have to press the “Run” arrow on either the “Block Diagram” or on the “Front Panel”.

5/11/2011
LabVIEW does “Generating Intermediate Files”. This files will be send to the Xilinx Synthesis Tools. But this is not important for us as application developers.
LabVIEW – FPGA Running VIs

LabVIEW starting the "Compile Server".
LabVIEW – FPGA Running VIs
When the “Bitstream generation is complete” message appears and the server status is set to “Idle…” the Xilinx synthesis tools have done their job.
You get a “Successful Compile Report” where you can see the implementation details of your code. You have to Press “Ok”.
After you have pressed the “Ok” button your VI starts running on the FPGA target. It is indicated on your screen by the black “Run” arrow.
• Playing with the switches SW0 and SW1 on the SPARTAN starter board you will see they have an XOR function. The led LD0 will be ON when one of those switches is turned ON.

• The problem with this implementation is that when you stop the VI, the function is erased on the LabVIEW FPGA board.

• If you don’t want this effect you can download this VI to the FLASH of the Xilinx SPARTAN starter board.
For implementing your VI into the FLASH on the SPARTAN Starter board you have to do some things.

1. Run when loaded to FPGA on. For this you have to go to “Project Explorer”.
2. “Right Mouse click” on your FPGA target. Select the “Properties” option.
“Run when loaded to FPGA”.
Make sure you select this option. Then press the “OK” button.
LabVIEW – FPGA into “FLASH”
We must recompile the VI because we made a change. Do this by going to the “Project Explorer” view. Click with the “Right Mouse button” on the FPGA VI you have created for this project. Select the “Compile” option.
• **We** will see that there are some previous steps executed.

• **When** the “Successful Compile Report” shows up you have to press the “OK” button.

**Now we will download it to the Flash.**

• **For** this you have to go to the “Project Explorer” view and do a “right mouse click” on the FPGA VI you created.

• **Then** choosing the “Download VI to Flash Memory” option will start downloading it to the SPARTAN starter board flash.
LabVIEW – FPGA into “FLASH”

LabVIEW FPGA VI is downloaded to the Flash.

5/11/2011
1. Implementing a Boolean function in LabVIEW FPGA on the Xilinx SPARTAN Boards (LAB 01).

2. Embedding VHDL code in a Xilinx SPARTAN FPGA (LAB 02).
• The first things we have skipped because it is the same as in LAB01.

• In this LAB we are going to create a parity checker.

(Parity checking is a rudimentary method of detecting simple, single-bit errors in a memory system. In communications, parity checking refers to the use of parity bits to check that data has been transmitted accurately. The parity bit is added to every data unit (typically 7 or 8 bits) that are transmitted. The parity bit for each unit is set so that all bytes have either an odd number or an even number of set bits).

• We will put “LED0” of the Spartan3E Starter board on when the input of the checker system (by use of “SW0”, “SW1”, “SW2” and “SW3”) has even bits.
Step 1: Adding FPGA I/O
Step 2: Creating the VI

Do a “right mouse click” on the “FPGA Target” in the “Project Explorer”. From the pull down menu select “New” => “VI”.

![Project Explorer with right mouse click on FPGA Target]

5/11/2011
Step 2: Creating the VI

On the “block diagram” of this VI we have to put a “Timed Loop”
Step 2: Creating the VI

In this “Timed Loop” you have to place a “HDL Interface Node”
Step 2: Creating the VI

Now double click the “HDL Node”; you will get the following screen:

In the first screen we try to add the following 2 parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>inp</td>
<td>in</td>
<td>TF [ ]</td>
<td>4</td>
</tr>
<tr>
<td>outp</td>
<td>out</td>
<td>TF</td>
<td>--</td>
</tr>
</tbody>
</table>

**Type**—Defines the data type of the parameter. LabVIEW displays only a subset of the data types supported in FPGA VIs.

**Length**—Defines the size of the Boolean array if you select the Boolean array data type in the Type column.
In the code screen we replace “hdlnode” (name of the entity) to “even_parity”.

The field below “architecture implementation of even_parity is” has to been filled with the following VHDL rule:

\[ \text{signal } s1, s2, s3, s4, s5, s6, s7, s8 : std\_logic; \]
The field below the VHDL keyword “begin” has to been filled with the following VHDL code:

\[
\begin{align*}
\text{outp}(0) & \leq ((s1 \text{ or } s2) \text{ or } (s3 \text{ or } s4)) \text{ or } ((s5 \text{ or } s6) \text{ or } (s7 \text{ or } s8)); \\
s1 & \leq (\text{not } \text{inp}(3)) \text{ and } (\text{not } \text{inp}(2)) \text{ and } (\text{not } \text{inp}(1)) \text{ and } (\text{not } \text{inp}(0)); \\
s2 & \leq (\text{not } \text{inp}(3)) \text{ and } (\text{not } \text{inp}(2)) \text{ and } \text{inp}(1) \text{ and } \text{inp}(0); \\
s3 & \leq (\text{not } \text{inp}(3)) \text{ and } \text{inp}(2) \text{ and } (\text{not } \text{inp}(1)) \text{ and } \text{inp}(0); \\
s4 & \leq (\text{not } \text{inp}(3)) \text{ and } \text{inp}(2) \text{ and } \text{inp}(1) \text{ and } (\text{not } \text{inp}(0)); \\
s5 & \leq \text{inp}(3) \text{ and } (\text{not } \text{inp}(2)) \text{ and } (\text{not } \text{inp}(1)) \text{ and } \text{inp}(0); \\
s6 & \leq \text{inp}(3) \text{ and } (\text{not } \text{inp}(2)) \text{ and } \text{inp}(1) \text{ and } (\text{not } \text{inp}(0)); \\
s7 & \leq \text{inp}(3) \text{ and } \text{inp}(2) \text{ and } (\text{not } \text{inp}(1)) \text{ and } (\text{not } \text{inp}(0)); \\
s8 & \leq \text{inp}(3) \text{ and } \text{inp}(2) \text{ and } \text{inp}(1) \text{ and } \text{inp}(0); \\
\text{enable}_\text{out} & \leq '1';
\end{align*}
\]
Step 2: Creating the VI

In the tab “Execution Control” check the option “Single-Cycle Timed Loop Allowed” and click the “OK” button.
Step 2: Creating the VI

Now we place two “FPGA I/O Node’s” on the block diagram of your VI.

Build an array with “SW0”, “SW1”, “SW2” and “SW3” as inputs.

We connect the output of this “build array” function to the input of the “HDLNode” called “inp”.

Connect now “LED0” to “outp” of the “HDLNode”.

---

Image of the block diagram with labeled nodes and connections.
Step 3: Running the VI

Now go to the “Project Explorer”. Right mouse click on “FPGA Target” then select “Properties”.

![Project Explorer Screenshot]
Step 3: Running the VI

Be sure to check in this window “Run when loaded to FPGA” before compiling the VI.
Step 3: Running the VI

Go back to “Project Explorer”. Do a right mouse click on “FPGA_VI_VHDL”. Select Compiler and let the LabVIEW FPGA code compile.

When the compilation completed go to

“Project Explorer” and do a right mouse click on

“FPGA_VI_VHDL” and select

“Download VI to Flash Memory”
Now you can pull out the USB cable out of the Xilinx Spartan3E board and press the PROG button on this board.

You will see that the function is implemented in it.
Take away home

1st LabVIEW today is a very powerful tool in Science.

2nd Contains an easy way to embed VHDL code in FPGA’s.

3rd http://www.youtube.com/watch?v=-ulWxOyOfgM
Using LabVIEW FPGA with the Xilinx SPARTAN-3E XUP

Thank You for Your Attention