FPGA tutorial

Lecture 1
Monday 07.09.2015 – 14:00

Jochen Steinmann
Organisation

• Structure:

1. Solution of the last Exercise
2. Presentation (FPGA / electronics / Verilog)
3. Hands on tutorial

Please ask your questions when they rise up!
Organisation II

• Requirements for this course
  – none
  – has a basic level, that everyone can follow!

• Slides and skeleton programs
  – will be inside the Indico before the tutorial

• Solutions will follow after the tutorial
Organisation III

● Hands-On:
  - 20 Laptops with VIVADO installed
  - 10 BASYS 3 Boards
    • 2 Laptops share one Board

● Today two lectures (2 x 90 min.)
  - we will have 2 times Lecture and Hands-On divided by a coffee break
What this tutorial cannot do

- Improve your soldering skills
  - it's just software / firmware.
- Make you an electric engineer
  - electronics is more than FPGAs...
- Make you an FPGA “expert”
  - it needs much more time...
What this tutorial can do

• Help you understanding electrical engineers talking about FPGA / Programmable Logic
• Help you programming a basic firmware
• Get an overview of programmable logic
  – Problems and their solutions
  – Features and how they can be used
Motivation

Why do physicists need FPGA-Knowledge

- All Experiment readouts are based on FPGAs
- Understanding the processes inside the FPGA helps troubleshooting the experiment
- Distribution of work:
  - electrical workshop:
    - PCB layout and production
  - physicists:
    - Firmware, tests and commissioning
Motivation II

• Programmable Logic is state of the art
  – designing prototype electronics
  – allow “easy” reconfiguration
    • If you need a dual core processor, just implement it
    • If you don't need it – remove it
    • Decisions can be done, when the PCB is produced

• Firmware has to match the experiment
  – Physicists should know all about their experiment
  – They know all the timing issues, etc.
Historical Background

Let's talk about history!
There is always a historical background!
Early times of digital electronics
Why programmable ICs?

- Producing a Chip costs much more than producing a PCB
- Using one Chip on many PCBs increases Number of Chips and reduces Costs per Chip
- If a Chip is programmable, it can be used on many PCBs in different situations
- Producing one wafer costs ~ 10k€
  - Not included man power to design wafer & packaging
- Saving space on a PCB
History of Programmable Logic Devices

• 1960th
  - Fuse Configurable Diode Matrix

• 1971
  - Programmable ROM

• 1978
  - Programmable Array Logic PAL
    - Gate Array Logic GAL reprogrammable
    - Complex Programmable Logic Device CPLD In circuit programmable
      Configured during programming → keep configuration

• 1989
  - Field Programmable Gate Array FPGA

PLA Programmable Logic Array
  every boolean operation
  fixed logic gates and programmable interconnections (matrix)
  OTP (one time programmable)

Development process CLPD & FPGA is almost the same
Hand wiring

Replace by part of one
What are FPGAs?

Field Programmable Gate Array

• Simple:
  – Programmable Logic Devices

• More detailed:
  – Programmable Logic Devices
    • With lot of additional function-block (e.g. RAM, DSP)
    • Flexible configuration
    • Can host mikrocontroller
What means Field programmable?

- FPGAs can be programmed in circuit
  - soldered on the final PCB
  - the surrounding electronics can be operated as normal, just the FPGA is programmed.
  - just need a couple (5 wires) to program a board
- The early programmable logic needs special voltages to get programmed.
- Boards with multiple FPGAs can be programmed via one connection to the board
  - FPGAs can be daisy-chained
PL and HEP?

• Almost every experiment uses FPGAs!
• Many readouts at R&D experiments are realised with FPGAs, because they are flexible
• FPGAs (can) have a well defined timing
• Provide easy interfaces to
  – Memory – different types SRAM, DDR
  – Ethernet – 10, 100, 1000 Mbit/s
  – PCI – PCI Express
  – High Speed serial transceiver (combined up to 2.8 Tb/s Virtex7)

Improve Firmware inside the Experiment
PL and HEP?

- readout of new detectors / prototypes
- easy to adapt to current situation
- readout can be done using various interfaces
  - Ethernet, USB, etc.
- future detectors have integrated digitisation
  - only digital signal conversion
    - collecting data of one event
    - zero suppression
Firmware written by Physicists

- **HADES**
  - Entire DAQ
- **PANDA**
  - Readout prototypes
  - Computing nodes
- **T2K – ND280**
  - Trigger
  - Data Concentrators
  - DAQ
- **Double Chooz**
  - Trigger Generation

Designed & built by RWTH Aachen
Work flow during FPGA development

Physicists

Idea

Firmware tests (simulations)

Final Firmware

Firmware evaluation

Experiment

Engineering

PCB layout

PCB production

PCB commissioning

How it should be...
This course

- we will focus on the newest XILINX 7 series

- Digilent BASYS3 Board
  - But the development environment also allows powerful simulations

- sometimes, you can see the NEXYS3 board, it's the same FPGA but a bit more hardware
Kinds of FPGA

XILINX

7-Series

SPARTAN

ARTIX

KINTEX

KINTEX

VIRTEX

VIRTEX

Increasing Performance and Functionality

ALTERA

Lattice

Actel

Physics Institute III

RWTH Aachen University
Xiling flag ship – 7 Series

• Announced & Released 2012

• Virtex 7
  • starting @ ~ 2,500 US-$ / chip
  • ending @ ~ 40,000.00 US-$ / chip

• Artix 7
  • starting @ ~ 40.00 US-$ / chip
  • ending @ ~ 500.00 US-$ / chip

You should always know what you are doing! when designing and programming FPGA-Boards
Xilinx 7 series

Xilinx Unified FPGA Series

- Low Power
  - Low
  - Lower
  - Lowest
- High Performance
  - Highest
  - Higher
  - High

Virtex™7
Kintex™7
Artix™7
HowTo Program...

• **Describe** your logic / functions in HDL
  – we use VERILOG

• Hardware **Description** Language
  – no “programming and compiling”

• Integrated Design Environment
What language?

• Hardware Description Language

• We will use Verilog
  – Alternative: VHDL

• Why Verilog?
  – quite similar to C / C++
  – easier to learn
  – need less words to describe the hardware
What does HDL mean?

• Origin in development of Chips
  - HDL were used to describe the behaviour of the chip and the simulation stimulus
  - another language was used to create the netlists needed for production
  - At some point it has been decided to use one language for both processes

With Verilog we can write the behaviour and simulation stimulus using the same language!

We don't want to spent time on learning the language, we want to use the hardware!
How “programming” a FPGA?

No programming, just description!

- **Hardware Description (HDL)**
  - Describe what the hardware should do.

- **Syntax Check**
  - Check Syntax

- **Synthesis**
  - “Compiles” the design to transform HDL source into an architecture-specific design netlist. (connections)

- **Translate**
  - Merges incoming netlists and constraints into a Xilinx design file, connections are merged with timing

- **Map**
  - Fits the design into the available resources, tells the FPGA which gate structures should be used

- **Place & Route**
  - Places and routes the design to the timing constraints, decides which gate is used for which function

- **Programming**
  - Write configuration into the FPGA or configuration memory
Basic Digital Input

- Direct user inputs
  - Keyboard, buttons, rotary dials, switches
- BASYS 3
  - Buttons
  - Slide switches
Xilinx Artix 7 – BASYS 3

Buttons

User Switches
Basic Digital Output

• Todays state of the art:
  – Light Emitting Diodes

• BASYS 3
  – Single (green) LED
  – 7 Segment LED (red)
    • 7 LEDs in one housing with a common Anode or Cathode
      → on Wednesday

\[ \text{LED on} = \text{High Level} = 1 \]
\[ \text{LED off} = \text{Low Level} = 0 \]
7-segment display

- common anode
  - all leds share one anode
- common cathode

- typically all segments of all digits use one output
- multiplexing using the persistence of vision (1 kHz)

We will use this 7-segment display later in this course!
Status

• now you should be familiar with
  - switches
  - buttons
  - LEDs
  - 7-segment display
1st Project

- First template:
  - Control all LEDs with the corresponding Slide Switch
    - Switch n controls LED n 1:1

- First introduction to VERILOG
- Get used to VIVADO and try to set it up
Verilog I

• How do we describe the hardware?
  - Datatypes
  - Functions (called module in Verilog)
  - basic operators

• Distinguish between
  - **combinatorial** logic “hardwired”
    • this you are able to build easily with discrete gates
  - **sequential** logic “do this, when”
Verilog II - Datatypes

- **reg**
  - source of signal

- **wire**
  - acts as a connection
  - transports a signal

Data – Levels:
- 0 – OFF
- 1 – ON
- X – undefined
- Z – high impedance

Every signal needs **one source**!
External inputs can also be sources.

Appears when data is not initialised.

**Always do proper initialisation!**

Needed for input and output on the same pin.
Means, that the output driver of the pin is not connected.

**We will not use this!**

Multiple drivers are not allowed!
Verilog III – Datatypes

- multiple data-"lines" can be grouped into busses
  - compare to arrays
- length must be given
  - wire [MSB:LSB] test

```verilog
global [15:0] sw; / wire [15:0] led;
```

- access to single elements:
- access to multiple elements:
  - test[3:0] ← first 4 bits
  - test[7:4] ← second 4 bits

Verilog can join busses:
```verilog
output = { sw[8:0], sw[15:8] };
```
Verilog III – Datatypes

• Filling datatypes
  - single bits → 0, 1
  - multiple bits → [Width][Radix][Value] → 2'b01
    • 2 bit wide
    • binary
    • 01 filled

Width of number has to match wire or reg width!

• We can also use:
  - 'b → binary
  - 'h → hexadecimal
  - 'd → decimal

Verilog also supports:
  'b01 → automatic width matching
  'b0 → sets all bits to zero
Signals ↔ Pinout

- constraint file, which is board specific.
- Here the signal names are mapped to the FPGA Pins.
  - e.g.

```bash
set_property PACKAGE_PIN V17 [get_ports {sw[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]

set_property PACKAGE_PIN U16 [get_ports {led[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
```

There is one predefined constraints – file (Basys3_Master.xdc)

Note: '#' marks comments, to use the lines, just uncomment them. Do not change the XDC file, because this can cause malfunctions.
Verilog III - Module

• Logic can be encapsulated in modules
  - modules can be in different files

• There must be one top-module, which does the connection from the FPGA to the board
  - signal names here are given in the constraint-file
Verilog III - Module

• First module using:
  - 16 slide switches
  - 16 LEDs

module BND01top(
  input [15:0] sw,
  output [15:0] led
)

  // do some logic with sw and led
endmodule
Verilog III - Module

• How are submodules instantiated

```verilog
module simple_logic(
    input [15:0] switch,
    output [15:0] ledout
)

    // do some logic with sw and led

endmodule

module BND01top(
    input [15:0] sw,
    output [15:0] led
)

    simple_logic mylog01 ( .switch(sw), .ledout(led) );
    simple_logic mylog02 ( .ledout(led), .switch(sw) );

endmodule
```

Pay attention to the order!
Verilog IV – doing logic

• Basic logical operators:
  – OR bitwise 'b01 | 'b10 = 'b11
  – AND bitwise 'b01 & 'b11 = 'b01
  – NOT bitwise ~ 'b01 = 'b10
  – XOR bitwise 'b01 ^ 'b11 = 'b10

• Basic comparisons
  – equality logical ==
  – inequality logical !=
  – AND logical &&
  – OR logical ||

There are a lot more operators: >, >=, < etc.
I will introduce them, when they are needed.
Verilog IV – doing logic

• How to assign values to outputs
  – combinatorial logic:
    assign led[0] = sw[0] & sw[1];  sw0 and sw1
    assign led[0] = sw[0] | ~sw[1];  sw0 or not sw1
Starting VIVADO

Newest Development Software from Xilinx for 7-Series Development

There should be a Icon on the Desktop!  
simply double click it!

We are using the free WebPack-Version.  
This license supports the used FPGA, but  
not all features of VIVADO.

You might get a warning, witch notes:   
Problem parsing board_part file ...

This is related to faulty file.  
The software is working as aspected.

For this course, we are using version  
2014.2
Open existing Project

- Download BND01skel.zip from Indico
  - unzip and open the project

- Run Synthesis

- Run Implementation
  - do not open the design (it needs to much time)

- Generate the Bitstream

- Program the Board

You can find all items on the left or top side of your VIVADO window
Running all steps within VIVADO

Run Implementation

Generate Bitstream

Run Synthesis (F11)
Run synthesis on your project source files.

takes some time ....

after its done

Synthesis successfully completed.
Next
- Run implementation
- Open Synthesized Design
- View Reports

Don't show this dialog again

OK Cancel
Failures ?!

- If one step fails always have a look at the messages.
- Xilinx is quite verbose and most errors can be solved by having a look at the messages.

In this first project there should be no error!
If you want to program a board:
Connect to Hardware Server

Open Hardware Target

This wizard will guide you through connecting to a hardware target.

To connect to a remote hardware target, provide the host name and IP port of the remote machine on which the instance of a Vivado Hardware Server is running.

after connecting a BASYS3 use the local server

To continue, click Next.
Select Hardware

Select Hardware Target

Select a hardware target from the list of available targets, then set the appropriate JTAG clock (TCK) frequency. If you do not see the expected devices, decrease the frequency or select a different target.

<table>
<thead>
<tr>
<th>Type</th>
<th>Port</th>
<th>Name</th>
<th>JTAG Clock Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>xilinx_tcf</td>
<td>Digilent/210274552340A</td>
<td>15000000</td>
<td></td>
</tr>
</tbody>
</table>

Hardware Devices (for unknown devices, specify the Instruction Register (IR) length)

<table>
<thead>
<tr>
<th>Name</th>
<th>ID Code</th>
<th>IR Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>xc7a100t_0</td>
<td>13631093</td>
<td>6</td>
</tr>
</tbody>
</table>

VCSE server: localhost:60001
Hardware server: 134.61.15.234:3121
Summary

Open Hardware Target Summary

- Hardware Server Settings:
  - Server: 134.61.15.234:3121

- VCSE Server Settings:
  - Server: localhost:60001
  - Version: 20

- Target Settings:
  - Target: xilinx_tcf/Digilent/210274552340A
  - Frequency: 15000000

To connect to the hardware described above, click Finish.
Test Boards

• There are 10 BASYS 3 Boards for this course
  – Try it on BASYS 3.
  – You can also do a simulation
    • I will show later how to do this

Everyone should have a chance to test his / her firmware on the real hardware!

Please let the others also use the boards!
Program
for normal, there is no need to change these values
What's happening now...

same should happen with the other switches
How-to Save your Solutions

- VIVADO has an option to Archive the Project
  - Generates small ZIP file
  - If you want to re-use it later

Uncheck

Ensure this is checked
1\textsuperscript{st} Project – Summary

- Start VIVADO
- First knowledge about Verilog
  - module
  - datatypes & levels
  - logical operators
- connect inputs with outputs
  - using combinatorical logic
    - assign
  - constraint file (XDC)
2nd Project – doing Logic

- We want to extend the first project by a bit more logic.
  - Create a submodule for the 2nd Project

Now we want to do some logic

- LED0 = SW0 & SW1 ← AND
- LED1 = SW0 | SW1 ← OR
- LED2 = SW0 ^ SW1 ← XOR
- LED3 = ~LED0 ← NOT

LED4 – LED7 = SW4 – SW7

Hint: merge single Bits to a Bus!
Hands On

• Please start with the Hands On!
  - solutions will be shown right after the coffee break.