The FPGA Hello World Example

Martin Schoeberl
martin@jopdesign.com

August 4, 2006

1 Introduction

When one starts to use a new language or environment the first program written is usually the famous 'Hello World' example. What is the 'Hello World' program in hardware, in an FPGA? The smallest project that produces dynamic output is a blinking LED. We will show the steps for a blinking LED example using Altera's Quartus and the Cyclone board Cycore.

2 Design Flow

All your design files (VHDL files) make up a *project* in Quartus. A Quartus II project is defined in Quartus II with just three files: `projectname.qpf`, `projectname.qsf`, and `projectname.cdf`. Those project files are plain text files and can be edited with any editor[1].

2.1 Create a New Project

Start Quartus II and create a new project with:

1. **File – New Project Wizzard...**
2. Select a project directory and select a project name. The project name is usually the name of the top-level design entity. In our case `hello_world`
3. In the next dialog box the VHDL source files can be added to the project. As we have no VHDL files at the moment we will skip this step
4. We have to select the target device. Choose family **Cyclone** and depending on your board either select `EP1C6Q240C8` or `EP1C12Q240C8` as device

[1] Close your Quartus II project before editing the files

5. We leave the EDA tools settings blank

6. Press **Finish** at the summary window

2.2 Device and Pin Options

As the default settings in Quartus II for a device are **dangerous**[2] we specify more details for our target device:

1. **Assignments – Device** to open the device properties

2. Press the button **Device & Pin Options...**

3. **Important!** At the tab **Unused Pins** select **As input tri-stated**.

4. If you intend to configure a dspio board via the USB download utility (`USBRunner`) select **.rbf** at the tab **Programming Files**.

5. The **LVCMOS**, selected in tab **Voltage**, is the *better* IO standard to interface e.g. SRAM devices

6. Close the dialog box and the next with **OK**

2.3 The Hello World VHDL Example

Add a VHDL file to the project with **File – New...** and select **VHDL File**. Enter the code from Figure 1 and save the file with filename `hello_world`.

2.4 Compiling and Pin Assignment

The analysis, synthesize, map, and place & route processes are all started with **Processing – Start Compilation**. The compiler did not know that our PCB is already done, so the compiler assigned any pin numbers for the inputs and outputs.

However, the pin numbers are fixed for the board. We have to assign the pin numbers for our two ports `clk` and `led`. Open the pin assignment window with **Assignments – Assign Pins**. Our two ports are already listed in the **All Pins** window. Double click the field under **Location** and select the pin number. Enter following assignments:

Node Name	Location
clk	PIN_152
led	PIN_166

[2] At the default setting the unused pins drive ground. However, some power pins from the EP1C12 are user pins in the EP1C6. The Cycore PCB can be used for both devices and therefore provides power on those pins. Driving them with ground (in the EP1C6) results in a short circuit between the ground driving pins and power.

```
--
--  hello_world.vhd
--
--  The 'Hello World' example for FPGA programming.
--
--  Author: Martin Schoeberl (martin@jopdesign.com)
--
--  2006-08-04  created
--

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity hello_world is

port (
    clk     : in std_logic;
    led     : out std_logic
);
end hello_world;

architecture rtl of hello_world is

    constant CLK_FREQ : integer := 20000000;
    constant BLINK_FREQ : integer := 1;
    constant CNT_MAX : integer := CLK_FREQ/BLINK_FREQ/2-1;

    signal cnt      : unsigned(24 downto 0);
    signal blink    : std_logic;

begin

    process(clk)
    begin

        if rising_edge(clk) then
            if cnt=CNT_MAX then
                cnt <= (others => '0');
                blink <= not blink;
            else
                cnt <= cnt + 1;
            end if;
        end if;

    end process;

    led <= blink;

end rtl;
```

Figure 1: The not yet famous VHDL 'Hello World' example

With the correct pin assignment restart the compilation with **Processing – Start Compilation** (or use the *play* button) and check the correct assignment in the compilation report under **Fitter – Pin-Out-File**. The `clk` pin should be located at 152 and `led` at 166; all unused pins should be listed as `RESERVED_INPUT`.

2.5 FPGA Configuration

Downloading your hardware project into the FPGA is called configuration. There are several ways an FPGA can be configured. Here we describe configuration via JTAG.

2.5.1 ByteBlaster

This section describes configuration via ByteBlasterMV connected to the printer port.

1. Connect your ByteBlasterMV to the printer port of the PC
2. Connect the other end to the JTAF header on the FPGA board. The red wire should point to pin 1 (written on the PCB)
3. Start the programmer with **Tools – Programmer**
4. Press the button **Auto Detect** – the programmer window should list two devices: an EPM2064A/7064AE and an EP1C6 or EP1C12
5. Double click for the filename of the EP1C6/12 device and select `hello_world.sof`. Leave the file blank for the EPM2064A/7064AE device
6. Select the checkbox under Program/Configure for the EP1C6/12 and press the **Start** button to configure the FPGA

The LED should now blink!

2.5.2 USBRunner

The dspio board can (also) be configured via the USB connection. In that case a short cable connects the JTAG header from the Cycore board with the JTAG header of the dspio board. The command line tool `USBRunner` needs a raw binary file (`hello_world.rbf`) for the FPGA configuration.

1. *TODO (Christof): Add a description of USBRunner and the change in the .cdf file – or change USBRunner*

3 Further Information

3.1 Tips

- The VHDL source files and the project files do not have to be in the same directory. Keeping them in separate directories is a better organization and simplifies reuse
- Use relative paths wherever possible to enable project sharing
- In general choose the entity name as filename. However, with different files (and names) for different versions of an entity a project can be easily *configured* just with different Quartus II projects

3.2 Quartus File Types

The most important file types used by Quartus:

.qpf Quartus II Project File. Contains almost no information.

.qsf Quartus II Settings File defines the project. VHDL files that make up your project are listed. Constraints such as pin assignments and timing constraints set here.

.cdf Chain Description File. This file stores device name, device order, and programming file name information for the programmer.

.tcl Tool Command Language. Can be used in Quartus to automate parts of the design flow (e.g. pin assignment).

.sof SRAM Output File. Configuration for Altera devices. Used by the Quartus programmer or by `quartus_pgm`. Can be converted to various (or too many) different format. Some are listed below.

.rbf Raw Binary File. Configuration for Altera devices. Used by the USB download utility (`USBRunner`) to configure the dspio board via the USB connection.

3.3 Links

- Quarts II Web Edition – VHDL synthesis, place and route for Altera FPGAs
- Jam STAPL Byte-Code Player – FPGA configuration in batch mode (`jbi32.exe`)
- Cycore Schematic – The board used in this example
- DSPIO board – The home of the dspio extension (schematic and USBrunner)
- hello_world.pdf – This document