# A Simple NIOS II SOPC Builder Design

Martin Schoeberl
martin@jopdesign.com

August 8, 2006

## 1 Introduction

Altera's SOPC-Builder is a powerful tool to create complete systems containing a processor, peripheral devices and memory interfaces. We will show the steps for 'Hello World' example with NIOS on the Cyclone board Cycore.

## 2 Quartus 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 `simple`
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
5. We leave the EDA tools settings blank
6. Press **Finish** at the summary window

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

## 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. Selected in tab **Voltage LVCMOS**; it is the *better* IO standard to interface e.g. SRAM devices
5. Close it with **OK**
6. Press teh Button **Migration Devices...** and add the second device to have EP1C6 and EP1C12 as migration devices
7. Close it with **OK**
8. Close the dialog box and the next with **OK**

## 2.3 Pin Assignment

To avoid assigning each pin location manually we will import the pin assignments for the Cycore board:

1. Open the Pin Planner with **Assignments – Pins**
2. Import the assignments with **Assignments – Import Assignmnts...**
3. Select file `cycore.csv` and close with **OK**
4. The **All Pins** window should now contain all pins
5. The Pin Planner can now be closed

[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.

## 2.4 SOPC Build

A simple NIOS-II system can be built with the SOPC builder:

1. Start SOPC builder with **Tools SOPC Builder...** and select a system name (e.g. `nios_system`)
2. Change the clock to 20.0 MHz
3. add a NIOS, On-Chip Memory (4 KB), and the JTAG UART
4. Select **System – Auto Assign Base Addresses**
5. Uncheck **Simulation** on the **System Generation** tab
6. Press the **Generate** button
7. Press **Exit**
8. Add all generated VHDL files with **Project – Add/Remove Files in Project...**

## 2.5 The Top Level VHDL File

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 `simple`.

## 2.6 Compiling and Pin Assignment

The analysis, synthesize, map, and place & route processes are all started with **Processing – Start Compilation**.

Check the correct assignment in the compilation report under **Fitter – Pin-Out-File**. The `clk` pin should be located at 152; all unused pins should be listed as `RESERVED_INPUT`. Pin 81 should be a `VCCINT` pin (it is a user pin for the EP1C6, but a power pin for the EP1C12).

## 2.7 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.

### 2.7.1 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 JTAG header on the FPGA board. The red wire should point to pin 1 (written on the PCB)
3. Start the programmer with **Tools – Programmer**

```
--
--  simple.vhd
--
--  top level for SOPC/NIOS II experiments
--

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

entity simple is

port (
    clk             : in std_logic
);
end simple;

architecture rtl of simple is

component nios_system is
    port (
        signal clk     : in std_logic;
        signal reset_n : in std_logic
    );
end component nios_system;

begin

-- Instantiate the Nios II system entity generated by the SOPC Builder.
nios: nios_system port map (clk, '1');

end rtl;
```

Figure 1: The top-level VHDL file for the NIOS II system

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 `simple.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

For the evaluation version a message box titled **OpenCore Plus Status** opens. Keep it open and start the NIOS IDE.

## 2.8 NIOS II IDE

The NIOS development system comes with an adapted version of Eclipse for C/C++ development. Follow the tutorial and create the small version of the 'Hello World' program:

1. **File – New – C/C++ Application**
2. Select a name
3. Select your SOPC project in **SOPC Builder System**
4. Select **Hello World Small**
5. Press **Finish**
6. Right click on your project and select **Build Project**
7. Right click on your project and select **Run As – Nios II Hardware** to start the program

You should now see in the console: **Hello from Nios II!**

## 2.9 External SRAM

We will now add one external SRAM chip to the NIOS II design.

1. Remove the On-Chip Memory from the design
2. See details in the Quartus II manual, SOPC Volume, Section: Off-Chip SRAM & Flash Memory
3. Create a SRAM component and add it to the SOPC system
4. Adapt the top-level `simple.vhd`

Figure 2 and 3 show our new top-level design. This system now contains 512 KB memory and can run bigger applications. Try out the standard 'Hello World' example from the NIOS II IDE.

Addition of the second SRAM chip for a 32 bit memory system is straight forward. Just make sure that both chip's address and control lines are connected to the signals from the SOPC design.

```
--
--  simple.vhd
--
--  top level for SOPC/NIOS II experiments
--

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

entity simple is

port (
    clk         : in std_logic;
    --
    --  only one ram bank
    --
    rama_a      : out std_logic_vector(17 downto 0);
    rama_d      : inout std_logic_vector(15 downto 0);
    rama_ncs    : out std_logic;
    rama_noe    : out std_logic;
    rama_nlb    : out std_logic;
    rama_nub    : out std_logic;
    rama_nwe    : out std_logic

);
end simple;

architecture rtl of simple is

    signal int_res          : std_logic;
    signal res_cnt          : unsigned(2 downto 0) := "000";    -- for the simulation
    -- for generation of internal reset
    attribute altera_attribute : string;
    attribute altera_attribute of res_cnt : signal is "POWER_UP_LEVEL=LOW";

    signal byte_nena        : std_logic_vector(1 downto 0);
    signal address          : std_logic_vector(18 downto 0);
```

Figure 2: The top-level VHDL file for the NIOS II system with an external 16 bit SRAM

```
begin

--
--  internal reset generation
--  no extern reset, epm7064 has too less pins
--
process(clk)
begin
if rising_edge(clk) then
    if (res_cnt/="111") then
        res_cnt <= res_cnt+1;
    end if;
    int_res <= not res_cnt(0) or not res_cnt(1) or not res_cnt(2);
end if;
end process;

--
--  Instantiate our Nios II system
--
nios: work.nios_system port map (
    -- 1) global signals:
    clk => clk,
    reset_n => not int_res,

    -- the_tri_state_bridge_0_avalon_slave
    chipselect_n_to_the_ext_ram => rama_ncs,
    read_n_to_the_ext_ram => rama_noe,
    tri_state_bridge_0_address => address,
    tri_state_bridge_0_byteenablen => byte_nena,
    tri_state_bridge_0_data => rama_d,
    write_n_to_the_ext_ram => rama_nwe
);

    rama_nlb <= byte_nena(0);
    rama_nub <= byte_nena(1);
    -- A0 from the avalon interface is NC on 16-bit SRAM
    rama_a <= address(18 downto 1);

end rtl;
```

Figure 3: The top-level VHDL file for the NIOS II system with an external 16 bit SRAM (cont.)

## 3 Links

- Quarts II Web Edition – VHDL synthesis, place and route for Altera FPGAs
- Quartus II Development Software Handbook – includes the SOPC builder volume
- Cycore Schematic – The board used in this example
- cycore.csv – The pin definitions for the Cycore board
- hello_world.pdf – The FPGA Hello World Example