1 Introduction

This tutorial explains how DDR3 memory modules connected to Intel’s DE5 Development and Education board can be used with a Nios® II system implemented by using the Intel® Platform Designer tool. The discussion is based on the assumption that the reader has access to a DE5 board and is familiar with the material in the tutorial Introduction to the Intel Platform Designer Tool and the tutorial Intel FPGA Monitor Program Tutorial for Nios II.

The screen captures in the tutorial were obtained using the Quartus® Prime version 18.1; if other versions of the software are used, some of the images may be slightly different.

Contents:

• Example Nios II System
• The DDR3 SDRAM Interface
• Using the Platform Designer tool to Generate the Nios II System
• Integration of the Nios II System into the Quartus Prime Project
• Using the Clock Crossing Bridge IP Core
2 Background

The introductory tutorial Introduction to the Intel Platform Designer Tool explains how the memory in an FPGA chip can be used in the context of a simple Nios II system. For practical applications it is necessary to have a much larger memory. The Intel DE5 board contains two DDR3 SODIMM (Small outline dual inline memory modules) slots that can be used to expand the amount of memory available to the FPGA. To provide access to the DDR3 SODIMMs, the Platform Designer tool implements a DDR3 SDRAM Controller with UniPHY circuit that is compatible with both DDR3 and DDR3L memory modules. This circuit generates the signals needed to interface with the DDR3 SODIMMs. The DDR3 standard requires careful timing between the memory modules and the system, so the DDR3 SDRAM Controller with UniPHY circuit uses a reference clock signal to produce two clock signals: one for the system and one for the memory module.

3 Example Nios® II System

As an illustrative example, we will add the DDR3 SDRAM to the Nios II system described in the Introduction to the Intel Platform Designer Tool tutorial. Figure 1 gives the block diagram of our example system.

![Block Diagram of Example Nios II System](image)

Figure 1. Example Nios II system implemented on the DE5 board.
The system realizes a trivial task. Four toggle switches on the DE5 board, SW3 – 0, are used to turn on or off the four green LEDs, LED3 – 0. The switches are connected to the Nios II system by means of a parallel I/O interface configured to act as an input port. The LEDs are driven by the signals from another parallel I/O interface configured to act as an output port. To achieve the desired operation, the four-bit pattern corresponding to the state of the switches has to be sent to the output port to activate the LEDs. This will be done by having the Nios II processor execute an application program. Continuous operation is required, such that as the switches are toggled the lights change accordingly.

The introductory tutorial showed how we can use the Platform Designer tool to design the hardware needed to implement this task, assuming that the application program which reads the state of the toggle switches and sets the LEDs accordingly is loaded into a memory block in the FPGA chip. In this tutorial, we will explain how DDR3 SODIMMs on the DE5 can be included in the system in Figure 1, so that our application program can be run from the DDR3 SDRAM rather than from the on-chip memory.

Doing this tutorial, the reader will learn about:

- Using the Platform Designer tool to include a DDR3 SDRAM Interface for a Nios II-based system
- Interfacing components clocked by different frequency signals on the DE5 board

4 The DDR3 SDRAM Interface

The signals needed to communicate with the DDR3 SODIMMs are shown in Figure 2. All of the signals can be provided by the DDR3 SDRAM Controller that can be generated by using the Platform Designer tool.

![Figure 2. The DDR3 SDRAM signals.](image-url)
5 Using the Platform Designer tool to Generate the Nios® II System

Our starting point will be the Nios II system discussed in the Introduction to the Intel Platform Designer Tool tutorial, which we implemented in a project called lights. We specified the system shown in Figure 3.

![Figure 3. The Nios II system defined in the introductory tutorial.](image)

If you saved the lights project, then open this project in the Quartus Prime software and then open the Platform Designer tool. Otherwise, you need to create and implement the project, as explained in the introductory tutorial, to obtain the system shown in the figure.

The DDR3 controller requires you to specify the parameters of your particular DDR3 SODIMM for it to function correctly. A list of necessary parameters are given in Table 1. The tutorial provides the parameters for the K4B2G0846C-HCK0 SODIMM manufactured by Samsung Electronics and supplied with the DE5 at the time of writing. If you use a different memory module, you will have search your module’s datasheet for the parameters listed in Table 1.
To add the DDR3 controller, in the window of Figure 3 select Memory Interfaces and Controllers > Memory Interfaces with UniPHY > DDR3 SDRAM Controller with UniPHY and click Add. A window depicted in Figure 4 appears.
Set the Speed Grade to 2, the Memory clock frequency parameter to 800.0 MHz, the PLL reference clock frequency to 50.0 MHz, Rate on Avalon-MM interface to Quarter, and leave other settings in PHY Settings as default. Click Memory Parameters to show the window in Figure 5.
Figure 5. DDR3 Controller Memory Parameters Window.

Set the Memory vendor to Samsung, the Memory format to Unbuffered DIMM, the Memory device speed grade to 800.0 MHz, the Total interface width to 64, the Row address width to 14, and the Column address width to 10. In the Memory Initialization Options, set the Memory CAS latency setting to 11. Output drive strength setting to RZQ/7, ODT Rtt nominal value to RZQ/6, Memory write CAS latency setting to 8, and Dynamic ODT (Rtt_WR) value to RZQ/4. The other settings should be left at their default values. Click Memory Timing to get to the window shown in Figure 6.
Set the timing parameters to the values shown in Table 1 then click on Board Settings to get to the window shown in Figure 7.
Figure 7. DDR3 Controller Board Settings Window.

In the Board Skews sub-menu, check FPGA DQ/DQS package skews deskewed on board and FPGA Address/Command package skews deskewed on board then set the other board skew parameters according to the values in Table 2.
Figure 8. Platform Designer system with the new DDR3 Controller.
Make the following connections:

- Connect the `pll_ref_clk` port of `DDR3_Controller` to the `clk` port of `clk_0`.
- Connect `global_reset` and `soft_reset` ports of `DDR3_Controller` to `clk_reset` port of `clk_0`.
- Connect the `jtag_debug_module_reset` port of the Nios II processor to the `soft_reset` port of `DDR3_Controller`.
- Connect the clock input of the `nios2_processor`, `onchip_memory`, `switches`, `LEDs`, and `jtag_uart` to the clock output, `afi_clk` of `DDR3_Controller`.
- Connect the `data_master` and `instruction_master` ports of the NIOS II processor to the `avl` port of `DDR3_Controller`.

Double click on the `avl` base address of the `DDR3_Controller` and set it to `0x4000_0000`. Your system should now look similar to the one in Figure 9. Right-click on the Nios II processor component to get to the window in Figure 10. Set the Reset Vector memory and Exception Vector memory to `DDR3_Controller.avl` and press Finish to return to the window in Figure 9. Click on Generate HDL... > Generate to generate your system and then close Platform Designer.
Figure 9. Final Platform Designer system with DDR3 Controller.
6 Integration of the Nios® II System into the Quartus® Prime Project

Now, we have to instantiate the expanded Nios II system in the top-level VHDL entity, as we have done in the tutorial Introduction to the Intel Platform Designer Tool. The entity is named lights, because this is the name of the top-level design entity in our Quartus Prime project.

The new top-level entity is presented in Figure 11. The input and output ports of the entity use the pin names for the 50-MHz clock, CLOCK_50, pushbutton switches, KEY, toggle switches, SW, and LEDs, LED, as used in our original design. They also use the pin names DDR3A_A, DDR3A_BA, DDR3A_CAS_n, DDR3A_CK, DDR3A_CKE, DDR3A_CK_n, DDR3A_CS_n, DDR3A_DM, DDR3A_DQ, DDR3A_DQS, DDR3A_DQS_n, DDR3A_ODT, DDR3A_RAS_n, DDR3A_RESET_n, DDR3A_WE_n, and RZQ_4, which correspond to the DDR3 SDRAM signals indicated in Figure 2. The names are the ones cited in the DE5 User Manual and are included in the file called DE5.qsf. The QSF file is included in the design files for this tutorial.
// Implements the augmented Nios II system for the DE5 board.
// Inputs: SW3-0 are parallel port inputs to the Nios II system.
// BUTTON0 is the active-low system reset.
// Outputs: LEDR3-0 are parallel port outputs from the Nios II system.
// DDR3 ports correspond to the signals in Figure 2; their names
// are those used in the DE5 User Manual.
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;
ENTITY lights IS
  PORT (
    SW : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    BUTTON : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
    OSC_50_B7D, RZQ_4 : IN STD_LOGIC;
    LED : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    DDR3A_A : OUT STD_LOGIC_VECTOR (13 DOWNTO 0);
    DDR3A_BA : OUT STD_LOGIC_VECTOR (2 DOWNTO 0);
    DDR3A_CAS_n, DDR3A_RAS_n, DDR3A_WE_n : OUT STD_LOGIC_VECTOR (0 DOWNTO 0);
    DDR3A_RESET_n : OUT STD_LOGIC;
    DDR3A_CK, DDR3A_CK_n : OUT STD_LOGIC_VECTOR (0 DOWNTO 0);
    DDR3A_CKE, DDR3A_CS_n, DDR3A_ODT : OUT STD_LOGIC_VECTOR (0 DOWNTO 0);
    DDR3A_DM : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
    DDR3A_DQ, DDR3A_DQS, DDR3A_DQS_n : INOUT STD_LOGIC_VECTOR (7 DOWNTO 0));
END lights;
ARCHITECTURE Structure OF lights IS
  COMPONENT nios_system
    PORT ( 
      clk_clk : IN STD_LOGIC;
      reset_reset_n : IN STD_LOGIC;
      switches_export: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      leds_export : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
      memory_mem_a : OUT std_logic_vector(13 DOWNTO 0);
      memory_mem_ba : OUT std_logic_vector(2 DOWNTO 0);
      memory_mem_ck : OUT std_logic_vector(0 DOWNTO 0);
      memory_mem_ck_n : OUT std_logic_vector(0 DOWNTO 0);
      memory_mem_cke: OUT std_logic_vector(0 DOWNTO 0);
      memory_mem_cs_n : OUT std_logic_vector(0 DOWNTO 0);
      memory_mem_dm : OUT std_logic_vector(7 DOWNTO 0);
      memory_mem_ras_n : OUT std_logic_vector(0 DOWNTO 0);
      memory_mem_cas_n : OUT std_logic_vector(0 DOWNTO 0);
      memory_mem_we_n : OUT std_logic_vector(0 DOWNTO 0);
      memory_mem_reset_n : OUT std_logic;
      memory_mem_dq : INOUT std_logic_vector(63 DOWNTO 0);
      memory_mem_dqs : INOUT std_logic_vector(7 DOWNTO 0);
      memory_mem_dqs_n : INOUT std_logic_vector(7 DOWNTO 0);
      memory_mem_odt : OUT std_logic_vector(0 DOWNTO 0);
      oct_rzqin : IN std_logic);
END COMPONENT;

Figure 11. The top-level entity that instantiates the expanded Nios II system. (Part a)
BEGIN

-- Instantiate the Nios II system entity generated by the Qsys tool.

NiosII: nios_system
PORT MAP (  
clk_clk => OSC_50_B7D,  
reset_reset_n => BUTTON(0),  
switches_export => SW,  
leds_export => LED,  
memory_mem_a => DDR3A_A,  
memory_mem_ba => DDR3A_BA,  
memory_mem_ck => DDR3A_CK,  
memory_mem_ck_n => DDR3A_CK_n,  
memory_mem_cke => DDR3A_CKE,  
memory_mem_cs_n => DDR3A_CS_n,  
memory_mem_dm => DDR3A_DM,  
memory_mem_ras_n => DDR3A_RAS_n,  
memory_mem_cas_n => DDR3A_CAS_n,  
memory_mem_we_n => DDR3A_WE_n,  
memory_mem_reset_n => DDR3A_RESET_n,  
memory_mem_dq => DDR3A_DQ,  
memory_mem_dqs => DDR3A_DQS,  
memory_mem_dqs_n => DDR3A_DQS_n,  
memory_mem_odt => DDR3A_ODT,  
oct_rzqin => RZQ_4);

END Structure;

Figure 11. The top-level entity that instantiates the expanded Nios II system. (Part b).

Perform the following:

• Enter the code in Figure 11 into a file called lights.vhd. Add this file and the nios_system.qip file produced by the Platform Designer tool to your Quartus Prime project.

• Import the pin assignments from the QSF file included in the design files for this tutorial.

• Perform analysis and synthesis of the design by clicking Processing > Start > Analysis and Synthesis

• Click Tools > Tcl Scripts... to open the window in Figure 12. Select and run the script Project > nios_system > synthesis > submodules > nios_system_DDR3_Controller_p0_pin_assignments.tcl; the script is required to correctly set the differential pins needed by the DDR3 SDRAM Interface.

• Compile the project.

• Use the Intel FPGA Monitor Program, which is described in the tutorial Intel FPGA Monitor Program Tutorial for Nios II to download and test the system on the DE5 board. Use the assembly program from the tutorial Intel FPGA Monitor Program Tutorial for Nios II to test the system; it has been reproduced for you in Figure 13.
If successful, the lights on the DE5 board will respond to the operation of the toggle switches.

```
equ Switches, 0x00002010
.equ LEDs, 0x00002000
.global _start
_start:
    movia r2, Switches
    movia r3, LEDs
loop:
    ldbio r4, 0(r2)
    stbio r4, 0(r3)
    br    loop
```

Figure 13. Assembly language code to control the lights.

7 Using the Clock Crossing Bridge IP Core

A clock crossing bridge allows components clocked by different frequency clock signals to interface and work with each other. This allows you to have low-speed and high-speed components in the same system without compromising the performance of your high-speed components. We will now modify our Nios system so that the human interface components (LEDs and switches) are run by a 50 MHz clock and the other components are run by the clock generated by the DDR3 SDRAM controller component.

Add the clock crossing bridge component to your Platform Designer system by selection Basic Functions > Bridges and Adaptors > Memory Mapped > Avalon-MM Clock Crossing Bridge. The window in Figure 14 will appear. Accept the default settings and press Finish. Right-click on the component and rename it Clock_Crossing_Bridge.
Make the following changes to your Nios system:

- Remove the connections from the data_master port of nios2_processor and the s1 ports of switches and LEDs.
- Connect the m0 port of Clock_Crossing_Bridge to the s1 ports of switches and LEDs.
- Connect the data_master port of nios2_processor to the s0 port of Clock_Crossing_Bridge.
- Connect the clk port of clk_0 to the s0_clk port of Clock_Crossing_Bridge and the clk ports of switches and LEDs.
- Connect the afi_clk port of DDR3_Controller to the m0_clk port of Clock_Crossing_Bridge.
- Connect the jtag_debug_module_reset of nios2_processor to the m0_reset and s0_reset ports of Clock_Crossing_Bridge.
- Connect the clk_reset port of clk_0 to the m0_reset port of Clock_Crossing_Bridge.

Double-click on the base address of Clock_Crossing_Bridge and set it to 0x1400. Similarly, set the base address of switches to 0x0000 and LEDs to 0x0010. The Nios processor will access these components at the address:
bridge base address + component base address. For switches this will be address 0x1400 and for LEDs this will be 0x1410. The complete system is shown in Figure 15. Regenerate the system then recompile the top-level VHDL entity. Finally use the updated assembly program given in Figure 16 to test your system with the Intel FPGA Monitor Program.

```
.equ Switches, 0x00001400
.equ LEDs, 0x00001410
.global _start
_start:
  movia r2, Switches
  movia r3, LEDs
loop:
  ldbio r4, 0(r2)
  stbio r4, 0(r3)
  br loop
```

Figure 15. The final Nios II system.

Figure 16. New Assembly language code to control the lights.
Copyright © Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Avalon, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel’s standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.