Nexys3 GPIO/UART Demonstration Project

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Revision: March 13, 2013

Functional Description

The Nexys3 GPIO/UART demonstration project provides a simple usage example of the Nexys3's GPIO and UART in a Xilinx Embedded Design Kit (EDK) design. It utilizes a UART controller, interrupt controller, timer, 3 GPIO cores, and a custom seven segment controller core that are connected to the Nexys3's LEDs, Switches, Push Buttons and Seven-Segment display. The Timer is configured to trigger an interrupt every second that increments the Seven-Segment display. The Push buttons are configured to trigger an interrupt that causes a message to be sent across the UART, and that also causes a corresponding digit in the 7-Segment display to be blanked.

This project demonstrates how to accomplish the following tasks in an EDK design that targets a Nexys3:

- Using a uartlite core as stdout
- Connecting a GPIO core to an interrupt controller
- Connecting a timer core to an interrupt controller
- Polling GPIO cores with simple memory access functions
- Using the custom seven-segment display controller (svn_seg_axi)

Hardware Implementation

**Note that this project was created using version 14.4 of Xilinx EDK. You may encounter problems if trying to use it with an earlier release.*

The system was designed using the Nexys3 BSB support package (available for download at www.digilentinc.com). We used the following system settings:

- AXI Based Design
- System Clock = 100 MHz
- Local Memory = 32 KB
- The following cores were included:
  - DIP_Switches_8Bits
  - LEDs_8Bits
  - Push_Buttons_4Bits
    - Interrupt Enabled
  - RS232_Uart_1
    - Xps_uartlite
    - 9600 Baud Rate
    - 8 Data bits
    - No parity
  - Digilent_SevSeg_Disp
  - Xps_timer
    - Width = 32
    - One timer is present
    - Interrupt enabled

- Llmb_cntlr and dlmb_cntrl

To view the resulting hardware configuration open project\system.xmp in Xilinx Platform Studio

Launching the Design in SDK

Follow these steps to launch the reference project on the Nexys3 using SDK (project file paths are given relative to the folder containing this document):

1. Open SDK and create a new workspace. It must be somewhere with no spaces in the name.
2. Under File, select New-> Application Project. A dialog box will open.
3. Under “Hardware Platform” select “Create New”. Another dialog box will open.
4. Under “Target Hardware Specification” click Browse. Select `.\source\hw_platform\system.xml` and press OK.
5. Click Finish to close the New Hardware Project dialog.
6. Name the project `gpio_demo`.
7. Ensure OS Platform is set to “standalone”, Language is set to “C”, and Board Support Package is set to “Create New”. Click Next.
8. Select Empty Application and click Finish.
9. Copy the three files from `\source\gpio_demo\` to `<Your_Workspace>\gpio_demo\src\`. You will have to overwrite the existing lscript.ld file there.
10. Build the workspace (if it is not already building automatically).
11. Obtain your Nexys3 and connect your computer to both the UART and USB PROG USB micro ports. Turn the board on and connect to it in a terminal program. Use a baud rate of 9600, 8 data bits, 1 stop bit, and no parity bit.
12. Under Xilinx Tools, select Program FPGA. In the dialog box that appears, ensure that “ELF file to Initialize Block Ram” is set as bootloop and click Program.
13. In the project explorer panel, expand gpio_demo -> Binaries. If possible, Right click on gpio_demo.elf and select Run As-> Launch on Hardware. If the “Launch on Hardware” option is not available, select “Run Configurations…” instead. In the dialog box, click “Xilinx C/C++ ELF” and then click the new button. Finally, click Run.
14. You should see each of the digits on the seven segment display counting from 0-9. Pressing any of the four outside buttons will darken a digit as long as it is held. Pressing the center button will trigger a user reset. The switches are all mapped to control their nearby LEDs. Also, if you observe the Terminal window you will see that a message is printed to it every time the push buttons trigger an interrupt. Try looking through the GPIO_demo.c and GPIO_demo.h source files to get a good idea on how it works.