Learning Objectives:

Project 2

- Gain experience designing an SoC with an embedded CPU
- Gain experience integrating hierarchical intellectual-property (IP) blocks into SoC design.
- Gain experience implementing an icon-based VGA video controller
- Gain experience writing assembly language for the PicoBlaze™ softcore CPU
- Learn SoC simulation (optional) and debug techniques

Project 2 demonstrations will be on November 3rd and 4th
Project 2 deliverables are due on D2L by Thursday, November 6th @ 10:00 PM

Project: RojoBot World

This project builds on the RojoBot concepts introduced in the Project 1 by placing the RojoBot in a virtual world. The RojoBot and its world are simulated in an IP (intellectual-property) block called *BotSim* (*bot.v*). BotSim is a SoC design in its own right; containing a Xilinx PicoBlaze, program memory and logic to manage its interface. You can treat BotSim as a "black box" for this project.

The BotSim module receives instructions that control the RojoBot wheel motors through a writable 8-bit "Motor Control Input" register. The BotSim provides information about its virtual environment through a set of read-only 8-bit registers. These registers indicate the RojoBot location, orientation (heading), direction of movement and the values of its proximity and line tracking sensors. The register details and function of the BotSim are described in the *BotSim 2.0 Functional Specification*.

You will be provided with a partial design example and documentation (*Proj2Demo*). The design example includes most of the Verilog code for a SoC design that instantiates BotSim, debounce, and 7-segment display logic and a Picoblaze CPU and memory for your application. The design example also includes PicoBlaze assembly language source code that demonstrates the basic function of the BotSim. You must add your own Verilog code to implement an I/O interface to the BotSim and to implement the top level of your design (nexys4fpga.v) before you can successfully run *Proj2Demo* on the Nexys4.

The I/O interface module you design connects the PicoBlaze application CPU to the BotSim registers and to the pushbuttons, switches, LEDs, and seven segment digits on the Nexys4 board. The module should also include interrupt logic for the PicoBlaze. The PicoBlaze interrupt hardware mechanism is described in the *KCPSM6 User Guide*.

Once you have the Proj2Demo design example running you can start on your Project 2 application firmware and hardware. From a hardware perspective you will implement an icon-based video controller that connects to a VGA monitor through the VGA connector on the Nexys4. From a firmware perspective you will implement a Picoblaze assembly language program that causes the RojoBot to traverse a virtual black line in its virtual world until it is stopped by a virtual wall.

Your PicoBlaze firmware will read the BotSim registers and drive its motor control inputs to direct the RojoBot to follow a black line defined in the RojoBot's virtual world. There are several algorithms to follow a black line. One of the simplest is to move forward until the RojoBot is no longer over the black line and reverse until it finds the back line again. After the RojoBot finds the black line again it should make a 45 degree turn to the right, move forward again until it loses the black line, and so on. This algorithm only works for a maze consisting of right turns so you will have to extend the algorithm to properly handle left turns, as well, without backtracking along the path it has already followed.

IMPORTANT: ALTHOUGH PROJECT 2 IS RELEASED AS A SINGLE PROJECT, IT IS OF SUFFICIENT COMPLEXITY THAT WE STRONGLY RECOMMEND THAT YOU IMPLEMENT THE PROJECT IN TWO PARTS. IN THE FIRST PART YOU IMPLEMENT THE HARDWARE AND SOFTWARE TO CONTROL THE ROJOBOT, MONITOR ITS PROGRESS THROUGH THE ROJOBOT WORLD AND TRAVERSE A RIGHT-TURN ONLY TRACK. PROGRESS CAN BE SHOWN ON THE 7-SEGMENT DISPLAY AND LEDS ON THE NEXYS4. YOU COMPLETE THE PROJECT IN THE SECOND PART BY IMPLEMENTING AN ICON-BASED VGA VIDEO CONTROLLER THAT VISUALLY SHOWS THE ROJOBOT PROGRESS AS IT TRAVERSES THE TRACK. YOU WILL DEMONSTRATE AND SUBMIT DELIVERABLES FOR THE VGA-ENABLED SOLUTION, ONLY.

Deliverables

- Demonstration of your project to one of the instructors or the T/A. In the demo we will expect your Bot icon to successfully traverse the assigned track. We will also expect the Bot icon to rotate to show its orientation (0°, 45°, 90°, 135°, 270° and 315°). Display will be on a VGA monitor.
- A *Theory of Operation* document (less than 10 pages). Your report should include a flowchart, state diagram or pseudo-code and text explaining your black line following algorithm.
- Source code for the Verilog module(s) you write. You do not need to include any simulation testbenches (simulation is optional)
- Source code for the PicoBlaze assembly language program(s) you write

Grading

There will be a single grade for this project based on the following rubric:

- (50 pts) A successful demonstration of correct operation on the Digilent Nexys 3.
- (20 pts) The quality of your *Theory of Operation* report where you explain your design.
- (15 pts) The quality of your design expressed in your Verilog source code. Please comment your code to help us understand how it works. The better we understand it, the better grade it can receive.
- (15 pts) The quality of your program expressed in well documented Assembly language code. The more readable your code, the higher grade it can receive.

IMPORTANT: Both team members will receive the same grade on the project regardless of the amount of or the quality of the work completed by each. You will be able to re-form teams for subsequent projects if you desire.

Project Tasks

1. *Download the Project 2 release and Picoblaze*

Download the project 2 release from the course website and the latest kcpsm6 release from the Xilinx website. The project 2 release .zip file contains Verilog HDL for BotSim and Proj2Demo, PicoBlaze assembly language code, and documentation. A description of the files is provided in *ECE 540 Project 2 List of Files*.

The Series 7 (for the Artix FPGA on the Nexys4 board) PicoBlaze release can be downloaded from the Xilinx website from

http://www.xilinx.com/ipcenter/processor_central/picoblaze/member/

2. *Create a new project in Vivado*

Open Vivado and create a new project. Add the following sources to the project.

- hdl_part1/bot.v
- hdl_part1/bot_pgm.v
- hdl_part1/map.v
- hdl_part1/world_if.v
- hdl_part1/world_map.v
- hdl_part1/proj2demo.v
- debounce.v (from Project 1)
- sevensegment.v (from Project 1)
- kcpsm6.v (from the PicoBlaze download)
- constraints/nexys4fpga_novideo.xdc

*kcpsm6.v is included in the Verilog directory of the PicoBlaze download from Xilinx.

3. *Add the RojoBot virtual world to your project*

Add world_map_part1/world_map.ngc to your Vivado project. world_map.ngc contains a pre-compiled netlist for the world map. It was created with the ISE Core Generator specifically for the Artix 7 FPGA on the Nexys4 board. world_map.ngc, along with world_map.v cause the netlist to be added to the project as a "black box".

4. *Create nexys4_bot_if.v and nexys4fpga.v . Add the new modules to your Vivado project*

Implement nexys4_bot_if.v to interface your PicoBlaze to the BotSim registers and Nexys4 debounce and 7-segment display modules. Don't forget to include the interrupt flip-flop. The interrupt input should be connected to the *upd_sysregs* signal from the BotSim. `kcpsm6_design_template.v` provides examples of how to instantiate the PicoBlaze, its program memory and several variations of interface modules.

nexys4fpga.v for this project can be derived from the module provided with Project 1, but you will need to make some changes. Refer to the *Proj2Demo Example Design Description* for more information on what modules it should instantiate.

Add both modules to your Vivado project.

5. (Optional) Simulate the system with ModelSim

You may want to build Verilog testbenches to simulate your system. If you include the PicoBlaze then you must also include the Xilinx simulation libraries in the model, as explained in the Xilinx documentation. You do not need to turn in testbench code or simulation results. The PicoBlaze HDL simulation capabilities are described in the *KCPSM6 User Guide*

NOTE: BECAUSE THE BOTSIM HAS A 50MS UPDATE TIME, IT WILL TAKE MANY, MANY THOUSANDS (POSSIBLY MILLIONS) OF SIMULATION CYCLES TO GET MEANINGFUL RESULTS.

6. Synthesize the demo system

Synthesize and implement your Vivado project. Check the warnings after synthesis for the usual suspects…mismatched port widths, signals that are not driven, signals that are tied to 0, signals driven by more than one source, etc.

7. Download the demo system to the Nexys4 board

If your Verilog nexys4_bot_if.v and nexys4fpga.v modifications are correct the programmed FPGA will respond to the pushbuttons and switches, illuminate the LEDs and display BOT activity on the seven segment display as described in *proj2demo.psm.*

8. Write and debug your new PicoBlaze program

Make a copy of, rename, and modify, *proj2demo.psm* to implement the black line following algorithm. You may choose to debug the program using a PicoBlaze instruction set simulator and assembler tool chain (http://mediatronix.com/pages/Tools). Test your black line following algorithm by simulating sensor values with and without the black lines and with and without obstructions. Be sure to "take ownership" of the code; this means updating the header, adding more comments, etc. In this course neatness and clarity in your source code are important.

9. Synthesize and implement the completed system using Vivado

Study the logs for synthesis errors and warnings. There may be some that could keep your design from operating correctly.

10. Download the system to the Nexys4 and test it

Verify that the display operates as specified in the functional specification. Verify that all buttons perform as specified. Verify that your RojoBot does, indeed, follow the black line to the ending wall by checking the performance of your Bot against the expected results.

Congratulations! You have completed the first part of the project.

11. Design, implement, and debug the RojoBot world video controller

The Project 2 deliverable includes a video interface to a VGA-compatible display. Instead of trying to follow the seven segment display to check that your Bot is navigating the track correctly you will be able to see your Bot move around the Bot World….cool, huh! The write-up for the video controller is in *RojoBot World Video Controller*.

References

[1] *Picoblaze for Spartan-6, Virtex-6 and 7-Series (KCPSM6) User Guide, Release 9* by K. Chapman.
[2] `kcpsm6_design_template.v` Included in the Picoblaze download from Xilinx
[3] *Xilinx Vivado Software Manuals*
[4] *BotSim 2.0 Functional Specification (Revision 2.1),* by Roy Kravitz
[5] *BotSim 2.0 Theory of Operation (Revision 2.1),* by Roy Kravitz
[6] *Proj2Demo Example Design Description (Revision 2.3),* by Roy Kravitz
[7] *RojoBot World Video Controller (Revision 2.1),* by Roy Kravitz
[8] *ECE 540 Project 2 List of Files* (Revision 2.1), by Roy Kravitz

<finis>