Learning Objectives:

Project 1

- 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 1 demonstrations will be on Jan 29 and Jan 30
Project 1 deliverables are due on D2L by Tuesday, Feb 4th @ 10:00 PM

You will work in teams of two on this project

Project: RojoBot World

This project builds on the RojoBot introduced in the Getting Started project 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 (*Proj1Demo*). 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 before you can successfully run *Proj1Demo* on the Nexys 3.

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 Nexys 3 board. The module should also include interrupt logic for the PicoBlaze. The PicoBlaze interrupt hardware mechanism is described in the *KCPSM6 User Guide*. You will also need to create nexys3fpga.v (the top level module).

Once you have the Proj1Demo design example running you can start on your Project 1 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 Nexys 3. 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 drives 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 1 IS 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 THE TRACK. PROGRESS CAN BE SHOWN ON THE 7-SEGMENT DISPLAY AND LEDS ON THE NEXYS 3. 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. 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.

IMPORTANT: Due to the large number of students in the course your team will be assigned a date for the demo and given a choice of times (first come, first served). Demo date assignments will be made by random drawing. We realize that this means that some teams will have to get their project done earlier than others but it is simply not possible to schedule demos for every team on a single day.

- 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. In this course you sink or swim as a team

which has been a bone of contention in the past…but that's the way it's got to be; we are simply not able to stand in judgment in these (hopefully) rare cases. You will be able to re-form teams for subsequent projects if you desire.

Project Tasks

1. Download the Project 1 release and Picoblaze

Download the project 1 release from the course website and the kcpsm6 release from the Xilinx website. The project 1release zip file contains Verilog HDL for BotSim and Proj1Demo, PicoBlaze assembly language code, and documentation. A description of the files is provided in *Project 1 List of Files*.

The Spartan 6 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 ISE

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

- hdl/bot.v
- hdl/bot_pgm.v
- hdl/world_map.v
- hdl/map.v
- hdl/world_if.v
- hdl/debounce.v
- hdl/sevensegment.v
- hdl/proj1demo.v
- kcpsm6.v
- synth/nexys3fpga.ucf

kcpsm6.v is included in the PicoBlaze download from Xilinx.

3. Add the RojoBot virtual world to your project

Copy world_map/world_map.ngc file into your ISE synthesis folder. This .ngc file was created by Xilinx Core Generator and contains an instance of the block RAM containing the map of the Rojobot virtual world.

4. Create nexys3_bot_if.v and nexys3fpga.v . Add the new modules to your ISE project directory

Implement nexys3_bot_if.v to interface your PicoBlaze to the BotSim registers and Nexys3 debounce and 7-segment display modules. Don't forget to include the interrupt flip-flop. It 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.

nexys3fpga.v for this project can be derived from the module provided in the Getting Started project. Refer to the *Proj1Demo Example Design Description* for more information on what modules it should instantiate.

Add both modules to your ISE 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 the ISE project. Synthesizing the PicoBlaze may cause hundreds of benign warning messages. You can have ISE filter these messages so that you can concentrate on the more important messages about *your* HDL code. To suppress a message from subsequent runs of the ISE, do the following:

1. In the "Processes" pane, double-click on "Design Summary/Reports". This will open the Design Summary pane. Check Design Properties/Enable Message Filtering.
2. Select the Design Summary Report and click on Implementation State/warnings to bring up the warnings.
3. Right-click on any of the messages and select the following: *Filter All Instances of This Message* to filter out messages with the same library name and message number, regardless of the message text. Filter only messages specifically related to the Picoblaze and/or the Picoblaze program memories. All the other messages may provide important and useful information about your design.

Note that when you suppress a message, it does not mean the issue is fixed, it means only that you have chosen to ignore the suppressed messages.

7. Download the demo system to the Nexys3 board

If your Verilog nexys3_bot_if.v and nexys3fpga.v modifications are correct, then 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 *proj1demo.psm.*

8. Write and debug your new PicoBlaze program

Make a copy of, rename, and modify, *proj1demo.psm* to implement the black line following algorithm. You may choose to debug the program using the OpenPICIDE instruction level simulator and assembler tool chain (http://openpicide.org/content/about/). 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 the completed system using ISE

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 Nexys3 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 Project 1.

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

The Project 1 deliverable includes a video interface to a VGA monitor. 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 7* by Ken Chapman.
[2] `kcpsm6_design_template.v` Included in the Picoblaze download from Xilinx
[3] *Xilinx ISE Software Manuals* (ISE Design Tools→ Documentation →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] *Proj1Demo Example Design Description (Revision 2.1),* by Roy Kravitz
[7] *RojoBot World Video Controller (Revision 2.1),* by Roy Kravitz
[8] *Project 1 List of Files* (Revision 2.1), by Roy Kravitz

<finis>