Computation II - 5EIB0
Lab 3: Creating a Mealy machine v1.0

Diederik Markus Wouter Schoenmakers

In the previous lab you have created a Moore finite state machine, another way to structure a finite state machine is the Mealy machine. In this lab you will recreate the coffee vending machine from lab 2 as a Mealy machine. Like in the previous labs you will first write and test the logic before proceeding to integrating the AXI interface and running it on the PYNQ board.

Contents

1 Creating a project **2**

2 The IP block **3**

2.1 The logic 3

2.2 Logic Testbench 4

2.3 The AXI4Lite interface 4

2.4 AXI4Lite Testbench 4

3 Debounce module **6**

4 Testing on the board **7**

1 Creating a project

As in the previous labs you need to create a project to work in.

Task 1. Create a vivado project.

1. First create a new folder in the file explorer. It is recommended that you place this in the SharedWork folder. **Do not use spaces in the folder name.**
2. Open vivado, you can use the menu in the top left of the screen.
3. To create a project select "File" $\rightarrow$ "Project" $\rightarrow$ "New". A pop-up will be presented on the screen, go ahead and click "Next".
4. You can now choose a project name and location. On default vivado will create a folder with the project in your home folder, it is recommended that you change this to the folder you just made. Make sure you have selected "Create project subdirectory". Click "Next" to continue.
5. Vivado has several different types of projects available, you want to make sure that you have "RTL Project" selected as well as "Do not specify sources at this time", you will add them to your project later on. Once again click "Next" to proceed.
6. The next thing Vivado wants to know is what kind of hardware you will be using. In your case this is the PYNQ z2 board. In the "Boards" tab, select the "pynq-z2" board. Click "Next" and then "Finish".

2 The IP block

Once again you are going to put the state machine in an IP block. The input and output of this IP is the same as for the moore machine.

2.1 The logic

Start by creating the IP block

Task 2. Creating your own IP block

1. Create an IP block on the default location, named `coffee_mealy`. The IP should have one AXI4Lite slave interface with at least 4 registers.
2. Add a design source named `coffee_mealy.v` and make this the only active file (so disable `coffee_mealy_v1_0.v` and `coffee_mealy_v1_0_S00_AXI.v`). Set `coffee_mealy.v` as the top module aswell.

You have now created the IP block and added the module to the block. The next step is to implement the mealy machine. The state diagram is depicted in figure 1. The mealy machine has the same inputs and outputs as the moore machine however unlike the moore machine, the mealy machine has a output at a transition and not a state. This poses a problem as these transition are instantly, if you only put the `coffee` led on during the transition than it would blink once very fast. You would not be able to see this. This is solved by simply updating the led at a transition and than leaving it on/off when the transition is done.

Task 3. Implementation of the Mealy coffee machine.

1. Add the following ports to your `coffee_mealy.v` module: 1 bit wide input port `clk`, 1 bit wide input port `insert`, 1 bit wide input port `reset`, 2 bits wide input port `coins`, 1 bit wide output port `coffee` and 3 bits wide output port `state_display`.
2. Write the implementation of the final state machine as provided in fig. 1. Make a transition when the `insert` button is pressed and reset the FSM when `reset` is high. Only update the output at the positive edge of the clock and only use edge triggering of the clock signal. To make sure that you only make a transition when the `insert` is **pressed**, check whether the current value is diffrent from its previous value. **Do not use: always @ (posedge insert).**

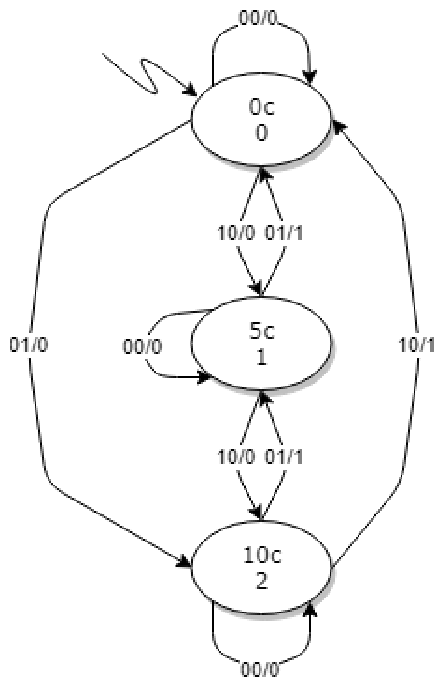

Figure 1: State diagram of the Mealy coffee machine.

2.2 Logic Testbench

To make sure that the logic works as intended you are going to create a testbench. This testbench should try all possible transitions in your final state machine just like in the previous lab.

Task 4. Write a testbench that tries all possible transitions in the state diagram and tries `reset` in every possible state. Simulate your testbench to verify that your vending machine works correctly.

2.3 The AXI4Lite interface

Your Mealy machine is now ready to be connected to the AXI4Lite interface. This is exactly the same as for the Moore machine so you can copy the code that you had made there.

Task 5. Connect the AXI registers to the input and output of your mealy machine.

1. Enable the automatic generated files you disabled in section 2.
2. Copy the changes you made in `coffee_moore_v1_0.v` and `coffee_moore_v1_0_S00_AXI.v` from Lab 2 to `coffee_mealy_v1_0.v` and `coffee_mealy_v1_0_S00_AXI.v` respectivly. Make sure that the output and input ports of the mealy machine are connected to input and output ports of your IP and that they are connected to the registers of the AXI interface. The registers of the AXI interface should all be read only aswell.
3. Change the top module to `coffee_mealy_v1_0.v`.

2.4 AXI4Lite Testbench

Since section 2.3 is praticly the same as for Lab 2, we expect you to be able to adapt it without the need to test it. When you are not sure if you did it correctly, you can use again Lab 1 to write the testbench for the

AXI4Lite interface.

Task 6. Synthesize your IP. In the bar on the left click on "Run Synthesis". Make sure that you resolve all errors and **critical warnings**. Vivado can synthesize a design with critical warnings however they often fail during implementation which means you will not be able to program the board. When Vivado is done synthesizing open the synthesized design do not run implementation.

Task 7. Package your IP block. Review Lab 1, if you do not know how to do it. When you are done switch back to the original project that you created in section 1;

3 Debounce module

Just like in Lab 2 you need to debounce the buttons before you can use them, you can copy the module you wrote for the moore machine.

Task 8. Copy `debounce.v` from the project of Lab 2 to your current project.

4 Testing on the board

Like in the previous lab you are not going to simulate with the microblaze and will instead be directly prgramming the board. To do this you need to setup a block design and write the C code. The block design is almost excactly the same as for the moore machine.

Task 9. Create a block diagram. The resulting structure should resemble figure 2.

1. Click on the left side of the screen under the top module "IP Integrator" on "Create block design". This will make a block design that you can use to connect the counter IP to the rest of the system. A new window is opened. Name the design `design_1`, and keep the rest of the settings untouched. Click "OK" to generate the block design. An empty window should be opened.
2. Include all necessary IP-blocks and modules and create the block design. Use block and connection automation to finish the block design.
3. Generate a wrapper for the block diagram. Right click the block diagram in the sources view and select "Create HDL Wrapper". Let vivado automatically update it.

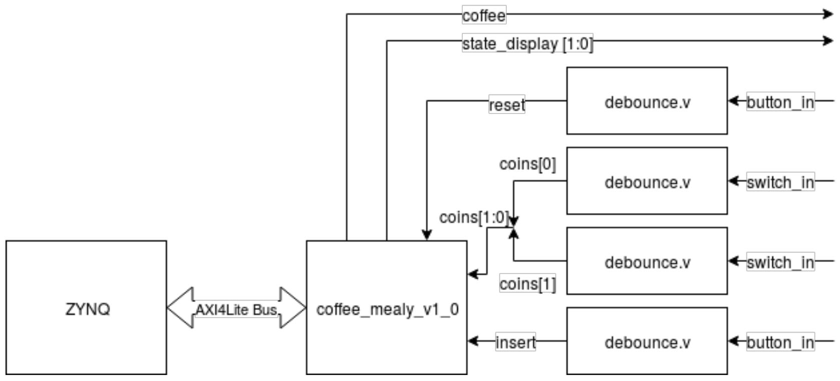

Figure 2: General structure of the block design.

Before you can generate the bitstream you need to setup up the constraints. This is excactly the same as for Lab 2.

Task 10. Connecting the external ports to physical pins.

1. Create a new **constraint** source ("file" → "Add sources"), select "Add or create constraints". Name this file `constraints.xdc` and place it local to the project.
2. Copy the contents from the constraints file you made in Lab 2. Check in the constraints wheter the wire names are still correct.

Task 11. Generate the bitstream. This will probably generate a lot of warnings, this is not unusual so can be ignored.

1. Click on "Generate Bitstream".
2. If you get a popup message asking to run synthesis and implementation click on "Yes".
3. In the next popup keep all the default settings and click "OK"

The C code for the mealy machine is the same as for the moore machine. You can copy the C code from Lab

2 however you do need to create a new project in the SDK.

Task 12. Launch the SDK and create a project.

1. Open the SDK by going to "File" → "Launch SDK".
2. In the newly opened SDK click on "File" → "New" → "Application Project". Set the name of your project to `fsm_code` and click on "Next", **not "Finish" like in Lab 1!**.
3. Select the hello world project and click on "Finish".
4. In the file `helloworld.c` you will find `int main()`. Delete the line `print("Hello World\n\r");`.
5. Program the FPGA and run the default code in the project. Verify that your FSM works correctly.

Task 13. Copy the C code form Lab 2. Check if the addresses stated the same. If that is not the case, change the C code. Test it on the board by uploading it to the FPGA. You can read the print statements in the SDK with the "SDK Terminal" on the bottom of the screen. Click on the green "+" and select `/dev/ttyUSB#`, leave the other settings at their default values.