# Lab4-8: Robot

## This lab is part of the final oral evaluation. You are expected to have answered all the questions below by yourself.

AI Warning: The use of Copilot, ChatGPT or any other AI assistant is not recommended. If you decide to persist in using those tools, you are fully responsible for its output, *therefore you need to be able to explain anything at the exam.*

Required: Lab3 must be completed.

## Warning

- Pico-ICE is susceptible to ESD. Handle the board with the PCB, not the connectors.
- IOs are unprotected. Do not create short circuits.
- IOs are NOT 5V tolerant, DO NOT HOOK ANYTHING 5V on this board.
- Oscilloscopes are earthed, connect only probe grounds on PCB ground pins.
- Always connect peripherals with all power supplies removed (including USB).
- ***Destroyed board will not be replaced during this lab if negligence is seen.***

## Specification Analysis

You will find the specification in Appendix A. Read it carefully and answer the following questions.

Explain:

- What functionalities are required to make this work?
- How would you store the configuration of the design?
- How do you connect interfaces?
- List the pins needed on the PMOD connectors.
- What interfaces are synchronous or asynchronous?

## Project structure

The package made of multiple directories:

- Open-logic-main: contains a library of useful design blocks.
- Project: Contains a ready to synthesize Radiant project.
- Rtl: Contains the source code you will write
- Sim: Contains some simulations to test the design
- Sw: contains extra Python code to test your design on the board

## Top-level

The top-level part of the design is in rtl/top. It is the base in which the new modules are instanciated. Keep the existing code except the part marked as “TO BE REMOVED DURING EXERCISES”.

Keep this pinout diagram https://pico-ice.tinyvision.ai/md_pinout.html handy, it will be needed for the future exercises.

Ensure QuestaSim provided with your installation is in your path. You should add c:/lscc/radiant/2024.2/questasim/win64 to your Path environment variable.

A simulation is available in the sim/top folder. You can launch it by executing “python run.py”. The simulator can show the gui using “python run.py --gui”. In this case you’ll need to type “vunit_restart” to relaunch the simulation without exiting Modelsim/QuestaSim.

If you have issues running your simulation, you can clean it by removing the vunit_out directory.

## Companion Software

Some Python SW is provided to do the Read/Write operations. It is in the sw folder. Use the the read/write functions of the PicoIce class. It handles the communication between your computer and the FPGA. See the main function for an example.

## USB to UART interface

The robot needs to be able to communicate with the outside world.

The UART1 of the Pico-ICE will be used to communicate with the FPGA over USB.

You will need a block that handles communication between CPU and FPGA.

Use an existing UART from open-logic project[1], the module’s name is olo_intf_uart.

Explain what the UART interfaces are and how your block is going to use them. One interface is AXI Stream[2]. Which signals go to FPGA pins?

The format of the serial frame is:

- *Write*: <HEADER=0xAA><Addr[7:0]><Data[15:8]><Data[7:0]>
- *Write answer*: <HEADER=0xAA><OK=0x0>
- *Read*: <HEADER=0x55><Addr[7:0]>
- *Read answer*: <HEADER=0x55><Data[15:8]><Data[7:0]>

From the description above explain:

- What is the address size?
- What is the data size?
- Describe the FSM states and transitions
- What is the APB required signals? What size are they?

Write an adapter between serial port and an APB master port.

[1] https://github.com/open-logic/open-logic
[2] https://developer.arm.com/documentation/ihi0051/latest

## Register Bank

The register bank stores the system's configuration.

Create an APB slave with the following address mapping:

| Address | Function |
|---|---|
| 0 | Red led (0: off, 1: on) |
| 2 | Green led (0: off, 1: on) |
| 4 | Blue (0: off, 1: on) |

Other registers will be added later in the lab sessions 😊.

Explain:

- Is an FSM required for the slave? Why?
- When is data actually read/written?

## HC-SR04 Controller

The HC-SR04 is an ultrasound distance sensor[3]. Read the datasheet.

You need to use olo_intf_sync to synchronize the pulse input.

Explain:

- How to generate a 10 µs pulse every 100 ms?
- How do you measure the pulse length?
- What is the maximum distance for a 16-bit register?

Write two modules, one for trig generation and one for echo measurement. Export the measurement value in the register file designed previously.

The Trig signal shall be connected to the pin ICE28 and the Echo signal shall be connected to pin ICE31.

## PWM Generator

The PWM is made with an up counter and a comparator. This will be used to control an L298N motor driver[4] using two outputs.

Explain:

- Which pins are required to control two motors?
- With a 12 MHz clock, what is the divider value to have a 25kHz PWM signal?
- How many bits are needed to represent the divider value?
- What is the minimum precision of the PWM signal?
- How do you handle the direction reversal with the L298N motor driver?

[3] https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
[4] https://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf

Write a module that generates the PWM and handles direction reversal for one channel. Instantiate it twice in the top-level.

The signals shall be mapped as follows:

| Signal | Pin |
|---|---|
| PWM_MOT1[1] | ICE44 |
| PWM_MOT1[0] | ICE45 |
| PWM_MOT2[1] | ICE46 |
| PWM_MOT2[0] | ICE47 |

The register bank shall add the following registers:

| Address | Register |
|---|---|
| 6 | PWM1 Direction [15],<br>PWM1 Speed [14:0] |
| 8 | PWM2 Direction [15],<br>PWM2 Speed [14:0] |

**You need common ground between the FPGA board and the motor board.**

## Quadrature decoder (Optional)

The quadrature decoder takes quadrature signals and gives a pulse count input that will be integrated.

Read this article[5].

Explain:

- How does a quadrature encode work?
- How to extract direction and position pulses?
- How to get speed indication?

Write the module that will do the quadrature decoding

The signals shall be mapped as follows:

| Signal | Pin |
|---|---|
| QUAD1[1] | ICE36 |
| QUAD1[0] | ICE38 |
| QUAD2[1] | ICE42 |
| QUAD2[0] | ICE43 |

The register bank shall add the following registers:

[5] https://www.fpga4fun.com/QuadratureDecoder.html

| Address | Register |
|---|---|
| 10 | QUAD1 Direction [15],<br>QUAD1 Speed [14:0] |
| 12 | QUAD2 Direction [15],<br>QUAD2 Speed [14:0] |

The Quadrature decoder shall also output the current speed as an AXI stream.

**Reminder: Those pins are NOT 5V tolerant. Ensure you have 3.3V input levels before connecting it to the board.**

**You need common ground between the FPGA board and the Quadrature decoders if they don't have the same power supply.**

## PI Controller (Optional)

The PI controller allows for close-loop control of a DC-motor. P and I terms need to be implemented.

Explain:

- What are the input values ranges?
- What is the Kp and Ki range?
- The dataflow graph of the equations?
- The pipelined version of the dataflow graph?

The register bank shall add the following registers:

| Address | Register |
|---|---|
| 14 | Kp |
| 16 | Ki |
| 18 | Set point speed |
| 20 | Enabled (0)<br>Use ramp generator (1) |

When enabled, the PWM uses the PI output as input value.

Write a module that implements the pipelined version of the dataflow graph.

## Ramp Generator

The ramp generator will generate acceleration and deceleration ramps for the motors.

Speed and Fast_time are variables. Time_delay, target_speed, speed_increment, speed_decrement are parameters stored in the register file.

It will follow this algorithm:

```
Speed = 0
while(Speed <= target_speed) {
   Speed += speed_increment
   If(speed >= target_speed)
      Speed = target_speed;
   Sleep(time_delay)
}
Fast_time = 0
while(Fast_time < Fast_run_time)
   Fast_time += time_increment
   Sleep(time_delay)
}
while(speed >= 0) {
   If (speed >= speed_decrement)
      Speed -= speed_decrement
   Else
     Speed = 0
   Sleep(time_delay)
}
```

Explain:

- How to translate this sequence into a VHDL design.

The register bank shall add the following registers:

| Address | Register |
|---|---|
| 22 | Time delay |
| 24 | Target Speed |
| 26 | Fast time |
| 28 | Speed increment |
| 30 | Speed Decrement |
| 32 | Execute Ramp (0) |

Write a block that will do the sequence defined above.

## Appendix A: Robot Specification

- The Robot shall have 2 DC motors interfaced with a dual full H-bridge.
- The Robot shall be able to move forward and turn around (same or opposite speed on both wheels)
- The Motor shall be controlled by a PI controller.
- The robot will be able to drive forward and backwards.
- The robot shall use quadrature encoders to measure distance
- The H-bridge PWM frequency shall be 40 kHz.
- The Robot shall be equipped with 2 HC-SR04 ultrasound sensors.
- The ultrasound sensors shall measure the distance every 500 ms.

- The Robot shall use the USB to UART to communicate with a host computer.

## Appendix B: FPGA requirements

- FPGA shall have a 12 MHz input clock
- All Asynchronous inputs shall be resynchronized
- The two ICE PMODs shall be used (see specification here https://digilent.com/reference/_media/reference/pmod/pmod-interface-specification-1_2_0.pdf)