# Project

## Aim

To become familiar with the design of digital circuits using a hardware description language, to simulate these designs and finally implement them using a FPGA.

## Assessment

This project has a weighting of 20% of the total subject marks.

Assessment is based on demonstration in the laboratory and submission of a written project report. The report is due at the end of week 12 with the video demonstration.

## Preliminary

It is assumed that the students have completed the “Xilinx ISE introduction” exercises. The second of these exercises may be completed during the 1st of the three lab sessions devoted to the project. It would be desirable to do some of this exercise beforehand.

These exercises are available in electronic form on Canvas.

## Introduction

The laboratory makes use of a FPGA prototyping board that plugs into the digital board that you have already been using and makes use of the existing facilities of that board. It is expected that a significant part of the implementation and testing would be conducted by simulation with actual hardware testing kept to a minimum.

**!!!Warning!!!**

When connecting the FPGA board connect the cables in the following order:

1. **Connect the JTAG Cable to the USB-JTAG Programming Module then the FPGA board.** Be very careful when connecting the JTAG cable plug - Do not connect it backwards as it may damage the board. If the Board is damaged, you would have to pay for it. The unit cost roughly USD200.
2. **Connect the micro USB to the USB-JTAG Programming Module and to the PC USB port last.**

**Reverse** this procedure when **disconnecting**.

**Exercise**

This design exercise is the creation of a controller for a rail - traffic intersection. The controller is responsible for controlling the traffic and walk lights for the intersection. You will use the **Motor** **and** **Traffic** **Modules** on your lab kit for this exercise.

The operation of the intersection is described by the following:



Train

Car

N

Ped

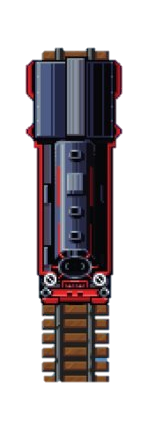
Ped

**Not Used**

Traffic Light

Traffic Light

Ped Light



* The rail track is from East to West (Right to Left)
* The traffic road is from North to South (Top to Bottom)
* The Pedestrian crossing is only on the North-South side (Top to Bottom)
* The East-West Pedestrian is NOT in use.
* Roads are a two-way street, cars are travelling from North to South and South to North.
* Initially, the North-South traffic has had green traffic light for a long time. Both traffic lights are green.
* When the train is arriving. (press the train button)
  1. Both Traffic lights will flash Red and Amber, with a 0.25s interval or 4hz.
  2. The Motor module will rotate Clockwise to simulate the Gate Barrier closing. The gate is fully closed after 5 rotations.
  3. After 30 seconds it is assumed that the train has fully crossed the intersection.
  4. The lights will continue to flash and the gate barrier will remain down.
* After 30 seconds has passed since the gate is down, if the North-South traffic is detected. (press the car button)
  1. The Motor module will rotate Counterclockwise to simulate the Gate Barrier opening. The gate is fully opened after 5 rotations.
     + This process can be interrupted if the train is detected. If so the barrier will rotate clockwise again to close the barrier.
  2. Once the Barrier is fully opened, both traffic will change to Green.
  3. The traffic will remain Green
* If the North-South Pedestrian is detected. (press the Ped button)
  1. The system will remember the button being pressed.
  2. After 30 seconds has passed since the gate is down, the Motor module will rotate Counterclockwise to simulate the Gate Barrier opening. The gate is fully opened after 5 rotations.
     + This process can be interrupted if the train is detected. If so the barrier will rotate clockwise again to close the barrier.
  3. Once the Barrier is fully opened, both traffic will change to Green, along with the North-South Pedestrian.
  4. The Ped Light will remain Green for 20 seconds, to allow for the pedestrian to pass. The system will then forget the button press.
     + If at any moment during this 20 second period, a train is detected, the system will prioritise the train, and start the process of closing the barrier and flashing the traffic lights.
  5. If no train is detected after 20 seconds, the pedestrian light is turned off, and the traffic green remains on.
  6. The system will remain in this state. Until a train or pedestrian is detected.

## Suggested Components

Timer – A counter with multiple outputs to represent various time delays. The counter would have a synchronous clear input to re-start the time interval.

State Machine – Provides the ‘intelligence’ of the system.

Controls:

* The Timer (Mealy-style outputs)
* The Lights (Moore-style outputs)

Responds to

* The Timer outputs
* The Car buttons (synchronised).
* The Pedestrian buttons (synchronised & registered).

## Provided Files

**Skeleton top-level VHDL file and UCF file**

You are provided with a skeleton top-level VHDL file for the project along with the corresponding UCF file to map the top-level pins to convenient I/O points on the FPGA module.

**Notes:**

* Refer to the earlier part of the manual for a description of the traffic intersection module input and outputs.
* The design should be divided into suitable VHDL modules. It would be expected that there would be separate modules for the state machine and counter at least.
* The clock for the system should be at least 100 Hz so that the system will be responsive to brief button presses.
* All synchronous elements of the design should operate on the same clock.
* To prevent input race conditions you must have single-point synchronisation of external inputs before using them as inputs to a synchronous circuit.
* There must be no gated clocks (check synthesis report ISE).
* There must be no latches (check synthesis report from ISE).
* An asynchronous reset signal should be used to reset all synchronous circuits (input synchronization, counters and the state machine) when the board is initially powered up. It should not be used to restart the timer as part of the usual circuit operation!
* Asynchronous reset or set operations should not be otherwise used!
* It is possible to adopt many different styles when writing state machine descriptions using VHDL. However, for this exercise, you are required to use the two-process style described in lectures.

## Project Submission Guidelines

The submission should include:

* Cover page including:
  + Subject name & number
  + Project title
  + Lab supervisor’s name
  + Group Member names
* Brief description of your solution to the problem. This would include:
  + English description of approach (max 1 page).
  + Block diagram of top-level structure.
    - Showing all modules with types and instance names
    - Signal names
    - Port names
  + State transition diagram (STD) or Algorithmic State Machine Diagram (ASMD) for the control block. It is advisable to use the simplified labeling of the STD (if used) to reduce clutter.
  + A detailed description of all the modules you have designed, highlighting its functionality.
  + A detailed description of the testing carried out (simulation and/or actual hardware)
    - Testbench
    - Simulation results
  + A detailed description on the debugging process. Bugs you found in your original codes and how you overcome it.
* Listing of VHDL modules.
  + These should be fully commented.
  + These **must** be printed out using a fixed-width font (a print from ISE is suitable).
  + These **must** be properly indented.
  + The use of well-chosen signal, port and module names is expected. The names should reflect the **purpose** of the signal. I would not expect to see **button1** on the state machine module (maybe ewPedButton)!
  + Well-chosen names in general e.g. state names such as **redNS**, etc rather than **S1**, **S2**.
* Attached the ISE Synthesis report.
* Refer to the project description for guidance on the required operation of the circuit and additional design points to check.
* Pay attention to the guidelines for VHDL and state machines given in lectures. Cryptic code and meaningless comments will LOSE marks.

Only a single submission **per group** is required.