Task 4

Finite state machines. EEPROMs designs

INTRODUCTION:

The main goal of this last task is to develop a project which includes two examples of digital control systems. On one hand, students will design the solution to a simple control system using a finite state machine (FSM). On the other hand, the same problem will be solved using an EEPROM memory to replace the combinational logic used in the first solution for both the excitation equations and the output equations.

The only sequential elements allowed in the design are the type-D flip-flops. The Xilinx ISE's "fd" component represents a type-D flip-flop. It is also possible to use a type-D flip-flop with a clear input, component "fdc".

During the assembly it is possible to use either the 74HC175 or the 74HC163. The later, which is a counter, will require to have the load input always enabled. This will make the counter to behave like 4-type D flip-flops.

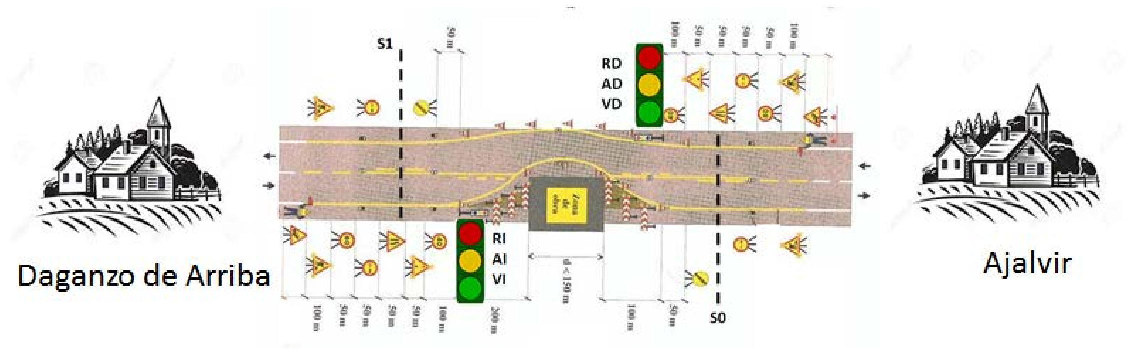
During the assembly exercise the utility of a ROM memory will be presented. The use of memories allows the implementation of complex logic functions. In this case, applied to a control system.

This task will have four sessions but a time span of five weeks. The first two session will be focused on Part 1. These sessions will take place on November 21st and 28th and on November 22nd and 29th. During the third week (December 5th and 6th), students will not have to attend to class. However, their task is to prepare the report corresponding to Part 1. This report will be considered for the evaluation of this part.

Important notice: Some of these exercises require preparations made prior to the beginning of the session. Please, read all the task and ensure you are correctly prepared for every session.

PART 1: DESIGN AND SIMULATION (FIRST AND SECOND SESSIONS)

The M-108 road, from Daganzo de Arriba to Ajalvir, is under reconditioning, allowing one car at a time. To reduce traffic jams, an automatic traffic light system is the best solution. The urbanism councilor from Ajalvir has asked her niece, who studies computer science at the EPS, to provide a Digital Control System using a finite state machine.



Exercise 1 (50%)

Propose, design and simulate a solution for the following traffic light control system. In the initial state, traffic should flow from Ajalvir (right) to Daganzo de Arriba (left). In this initial state, Ajalvir's traffic light will be on green (VD is '1') and Daganzo's traffic light will be on red (RI is '1'). The system will be in this state while no cars are detected by Daganzo's car sensor (S1 is '0').

From this initial state, when a car is detected by Daganzo's sensor (S1 is '1'), the system will keep the traffic light's values for two more clock cycles. On the third one, Ajalvir's traffic light will be switched to yellow (AD is '1' and VD is '0').

On the next cycle, Ajalvir's traffic light will switch to red (RD is '1') and Daganzo's traffic light will switch to green (VI is '1'). Then the system will keep this configuration until a car is detected by Ajalvir's sensor (S0 is '0'). The system will switch the traffic lights the same way as they were switched in the previous case. The left side traffic light will keep the green light on, and on the third cycle it will be switched to yellow for one clock cycle. During these three clock cycles, the right side traffic light will keep its red light on. On the fourth cycle, the left side traffic light will turn its red light on and the right side will turn its green light on. This state corresponds to the initial state.

The system will include a CLEAR input, active on high level, to reset the finite state machine.

To fulfill the simulation, students will have to run the simulation for 10,000 ns (right click on "Simulate Behavioral Model" → Properties → Simulation Run Time 10000),

The FSM to be designed will have:

- 4 inputs (CLK, CLR, S0, S1)
- 6 outputs (RD, AD, VD, RI, AI, VI)

Create a Moore FSM, design it using Xilinx ISE and simulate this design using ModelSIm. The only sequential components that can be used in this design will be type-D flip-flops (fd component in Xilinx ISE). If a clear input is required, you may use the fdc component in Xilinx ISE. To verify the correctness of the design using simulation, a is provided. This set consists of the following file: "p4ej1_tb.vhd". To ensure that the testbench correctly works, the following conditions must be fulfilled:

- Component name: p4ej1
- File name for schematic: p4ej1.sch
- Input names: CLR, CLK, S0, S1
- Output name: RD, AD, VD, RI, AI, VI

Exercise 2: (50%)

To simplify the final implementation and reduce costs, a new design has been requested. This new design will solve the same problem as in exercise 1 using type-D flip-flops and memory blocks.

To store M different N-input equations the following memory block is required:

- Address width = N
- Data width = M

This problem defines seven equations (AI, AD, VD, VI, D0, D1, D2) with 5 inputs (Q0, Q1, Q2, S0, S1) each. So the requirements define a 5-bit width address memory with 7-bit data width. An equivalent solution is to use seven 1-bit memories with a 5-bit width address. Xilinx ISE provides a memory with 32 different 1-bit words (component ROM 32x1).

As a design approach, create a different schematic including the 1-bit ROM memories that are required and combine them into a 32 N-bit memory. Use the new symbol to design the solution to the problem.

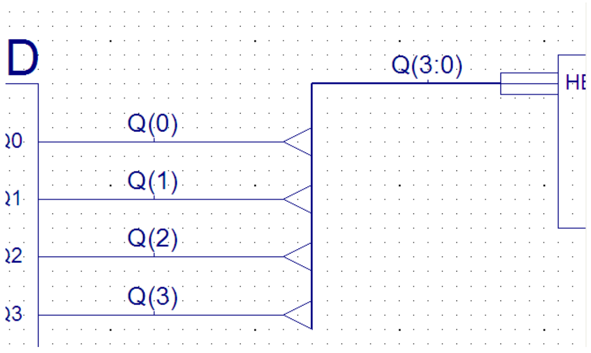
To define the contents of each of these memories, right click on each of the components and select "Object properties". The row INIT defines the memory's 32-bit content using a hexadecimal value. The most significant bit of this hexadecimal number corresponds to the content of the highest address position ("11111").

- Component name: p4ej2
- File name for schematic: p4ej2.sch
- Input names: CLR, CLK, S0, S1
- Ouput name: RD, AD, VD, RI, AI, VI

How to create a bus

The following figure shows how to create a bus. These are the steps to achieve this:

- Create a signal using "Add wire"
- Assign a name to this new signal "XXX(n:0)"
- Create a bus tap using "Add bus taps" for each point where an single bit signal needs to be connected to the bus.
- Connect single bit signals to the bus tap and name them "XXX(i)"



Submission

The submission deadline is Tuesday, December 11th at 23:55 for the Wednesday groups and Wednesday, December 12th at 23:55 for the Thursday groups. A member of each group must submit to Moodle the following three files:

1. A compressed file containing the following:
   a. Al the schematic files created during this task (.sch files)
   b. A PDF report describing all the steps that were followed to create the FSM: truth table, Karnaugh maps, simplified excitation equations and simplified output equations.
   c. Simulation screenshots obtained with ModelSim. These screenshots must include every input and output of the design. Also, a brief explanation of the result must be included.
1. Note: The name of this compressed file must identify the team and the group. E.g. the file “1101_5.zip” will be the submission of team 5 within group 1101.
2. A PDF file containing the assembly diagram created using the back annotate technique presented during tutorial.
3. The text file that will be used to record the memory.

Note: The name of this file must identify the team and the group. The file must have “.hex” extension. E.g. the file “1101_5.hex” will be used to program the memory for team 5 within group 1101.

PART 2: IMPLEMENTATION (THIRD SESSION)

Important notice: This exercise requires mandatory preparation prior to the beginning of the session.

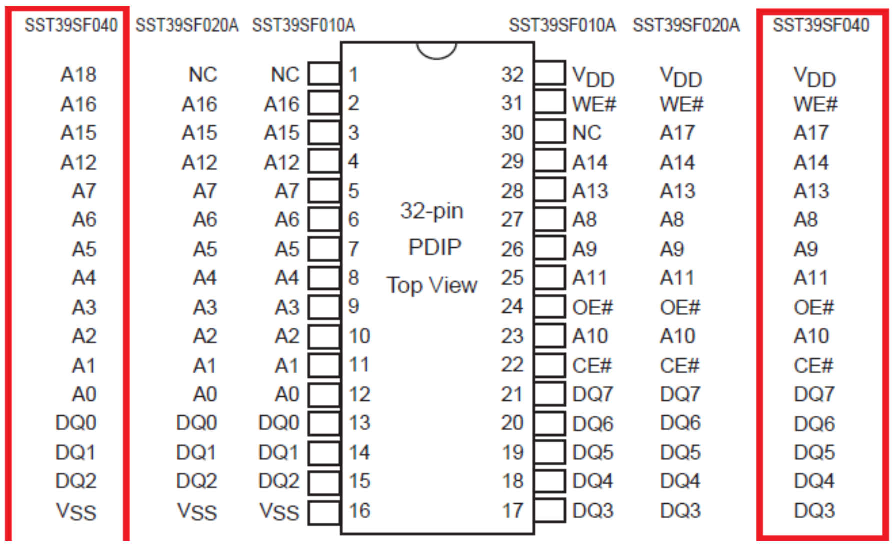
Implement the design corresponding to exercise 2 using an EEPROM memory and either the FF-D 74HC175 or the counter enabling the load input. Each valid position of the EEPROM must contain the next state value (the inputs for the FF-D / counter) and the output values for the current state. The system's clock signal (CLK) will be generated using the 1 Hz clock provided in the trainer. As an alternative, any of the trainer's switch or button may perform this task. It is also mandatory to include an active high CLEAR input signal (CLR). It will reset the FSM.

Input signals will be generated using trainer's switches and the output signals will be connected to the trainer's leds. Using labels is recommended for clarifying the assembly. All output signals must be connected to the leds.

Note: To debug any mistake in the assembly, it is recommended to connect CLK and CLEAR inputs, and the FF-D's outputs (current state) to the leds to obtain more information.

Using the EEPROM

This exercise requires using an EEPROM. The provided memory is SST39SF040. This is an electrically erasable programmable read only memory (EEPROM). It's capacity is 4 Mbit organized in 524,288 8-bit words. Therefore, it requires a 19-bits address bus and has an 8-bits data bus. The pinout of the chip is presented in the following figure (more information is provided in the data sheet).


SST39SF040
SST39SF020A
SST39SF010A
SST39SF010A
SST39SF020A
SST39SF040
32-pin
PDIP
Top View
A18 NC NC 1 | 32 VDD VDD VDD
A16 A16 A16 2 | 31 WE# WE# WE#
A15 A15 A15 3 | 30 NC A17 A17
A12 A12 A12 4 | 29 A14 A14 A14
A7 A7 A7 5 | 28 A13 A13 A13
A6 A6 A6 6 | 27 A8 A8 A8
A5 A5 A5 7 | 26 A9 A9 A9
A4 A4 A4 8 | 25 A11 A11 A11
A3 A3 A3 9 | 24 OE# OE# OE#
A2 A2 A2 10 | 23 A10 A10 A10
A1 A1 A1 11 | 22 CE# CE# CE#
A0 A0 A0 12 | 21 DQ7 DQ7 DQ7
DQ0 DQ0 DQ0 13 | 20 DQ6 DQ6 DQ6
DQ1 DQ1 DQ1 14 | 19 DQ5 DQ5 DQ5
DQ2 DQ2 DQ2 15 | 18 DQ4 DQ4 DQ4
VSS VSS VSS 16 | 17 DQ3 DQ3 DQ3

This chip has the following control signals: CE (chip enable), WE (write enable) and OE (output enable). These signals must be connected as follows: CE and OE must be connected to GND and WE must be connected to VDD. This last connection must be the first one to be made, in order to avoid accidentally programming the memory. Any unused address pin (Axx) must be also connected to GND. **The memory will not properly work if these instructions are not fulfilled**.

Programming the EEPROM

The first step to program an EEPROM is to define its contents. The programmer uses a specific format to define the values of each memory block. This format is a set of records, each of them has different fields:

- Record’s delimitation character “:”
- Record’s data field length (in bytes) as a two-digit hexadecimal number
- Memory initial address as a four-digit hexadecimal number
- Record’s type as a two-digit hexadecimal number. The only required values are:
    - 00 to specify a data record
    - 01 to specify a file end record
    - 02 to specify an extended address record (not used)
- Record’s data field, a two-digit data value. This field’s length is specified in the record data length.
- Checksum. The last field of the record is a two-digit hexadecimal number. Its value is defined so the sum of every two-digit number of the record (including the checksum) is 0x00.

Consider the following record as an example:

:**10**0080**00**AF5F67F0602703E0322CFA92007780C3**FD**

- The length field “10” states that the data field’s length is 16 bytes.
- The address field “0080” states that the initial address for the data is 0x0080. Therefore, the memory will be filled as follows:
- The record’s type field “00” states that this is a data record.
- The record’s checksum is “FD”

Consider the following record as a second example:

:**00**0000**01**FF

- The length field “00” states that the data field’s length is 0 bytes.
- The address field “0000” states that the initial address for the data is 0x0000.
- The record’s type field “01” states that this is an end of file record.
- The record’s checksum is “FF”.

This notation only allows a 16-bits address bus. Hence, it is not possible to define all the contents of the memory. Pins A16, A17 and A18 must be connected to GND so their value is ‘0’.

To define the content of the memory, consider the relation between the connections of the address pinouts to the FF-D outputs (Q2, Q1, Q0) and the switch inputs (S0 and S1). Also define which data pin corresponds to each output or the next state value (D2, D1, D0).

Important notice: To perform the assembly it is mandatory to have presented the assembly diagram the previous week. **The submitted report will contain the assembly diagram**, the list of required integrated circuits and the required connections (check tutorial on task 0b). **No team will be allowed to start the assembly if the assembly design is not presented.**

EVALUATION

The grade of Task 4 is obtained using the following formula:

$$Grade_{T4} = 0.35 \cdot Design + \ 0.3 \cdot Implementation + 0.35 \cdot Exam$$

- **Design:** the evaluation of the design will be based on the submission described in part one. completed during the first session of the task. When a group of students finishes an exercise, they must ask the teacher to evaluate it.
  To fulfill the evaluation, every group must submit the results of every exercise to Moodle.
  Important notice: Any submission not recorded in Moodle will be graded with a 0.

  The evaluation of the design will consider the following elements:
  - The correctness of logic function simplification
  - The clarity of the schematic design
  - The completeness of the simulation (every input combination should be tested)
- **Implementation:** the evaluation of the implementation will be performed during the second session of the task. When a group of students finishes an assembly, they must ask the teacher to evaluate it.
  Important notice: No assembly will be graded if the corresponding assembly diagram is not presented to the teacher.
  The evaluation of the assembly will consider the following elements:
  - The correctness and completeness of the implementation. Every input combination will be evaluated.
- **Exam:** During the first 30 minutes of the session that follows this task, an exam related to the designs of this task will take place. The purpose of this exam is to verify that the students have acquired the required knowledge about the concepts and designs presented during the task.