An undergraduate laboratory course on fuzzy controller implementation in FPGAs

S. Alcántara, C. Pedret, R. Vilanova and R. Moreno
Systems Engineering and Automatic Control Group
Autonomous University of Barcelona
08193 Bellaterra, Spain
Salvador.Alcantara@uab.es

Abstract—In this paper the guidelines for a lab course consisting of implementing a fuzzy controller in an FPGA are given. The aim is to introduce it at the Autonomous University of Barcelona as an undergraduate lab course on embedded control using FPGAs.

I. INTRODUCTION

Field Programmable Gate Arrays (FPGA) [1] have evolved a lot during the past few years. The tremendous increase in capacity together with the reduction in cost have made FPGAs to gain importance in the field of control as an embedded platform alternative to microcontrollers or other more traditional software-based solutions [2],[3]. The main advantages of FPGAs over general purpose computing chips like digital signal processors (DSP’s) or microcontrollers are their lower power consumption and faster speed of operation due to their inherent hardware parallelism, being they totally suited as hard real-time applications embedded solutions. However, few are the labs in undergraduate control courses which make use of FPGAs for design implementation. In this paper a FPGA-based implementation of a fuzzy logic controller (FLC) [5] is proposed as the laboratory work to accompany a new course covering technological aspects of control (embedded platforms for real-time control) as well as different control paradigms (fuzzy control, model predictive control, etc). More specifically, the laboratory project consists of the implementation of a two-input-one-output fuzzy controller for an inverted pendulum available in one of the department laboratories, see [7] and [8] for the theoretical background of fuzzy systems with regard to an inverted pendulum.

The main motivation for organizing the lab course is to provide the chance for students to get used to FPGA technology, due to its ever-increasing role in embedded applications (see [4] for an excellent textbook on FPGAs and embedded systems), and provide an opportunity to integrate and complement topics covered in previously taken courses in computer science, programming, controls, microcontrollers and electronics.

II. CONTROL CURRICULUM AND MOTIVATION

Our students are mainly computer and electrical engineers who course several subjects on digital systems (including FPGAs) during the first three years. Optionally, they can take some subjects on automatic control, where they deal with classical control theory, both continuous and digital. The aim of the present work is to provide the guidelines for a laboratory project to accompany a new course covering technological aspects of control (embedded platforms for real-time control) as well as different control paradigms (fuzzy control, model predictive control, etc). More specifically, the laboratory project consists of the implementation of a two-input-one-output fuzzy controller for an inverted pendulum available in one of the department laboratories, see [7] and [8] for the theoretical background of fuzzy systems with regard to an inverted pendulum.

The main motivation for organizing the lab course is to provide the chance for students to get used to FPGA technology, due to its ever-increasing role in embedded applications (see [4] for an excellent textbook on FPGAs and embedded systems), and provide an opportunity to integrate and complement topics covered in previously taken courses in computer science, programming, controls, microcontrollers and electronics.

III. THE LABORATORY SETUP

The system to work with is the classical inverted pendulum mounted on a cart. In figure 1 we can see a photograph of the actual system. The controller to be implemented will have to keep the pendulum on its upright position acting upon a dc motor connected to the pendulum belt, see figure 2.

The inverted pendulum provides sensing of angle deviation from the vertical and pendulum cart displacement through two potentiometers. Using a similar approach as described in [7], we take two states and one control variable. The first fuzzy variable is the angle and the second is the pendulum cart speed (it will be calculated from cart position). As output fuzzy variable we use the dc motor armature voltage.
VHDL is the hardware description language that will be used to entry the design, which is to be implemented mixing behavioral with structural descriptions, depending on the nature of each block of the design and optimization criteria.

During the lab, apart from the inverted pendulum system with its associated electronics, the following material will be available:

- A PC with a PCI data acquisition board (Advantech PCI-1710HG) to be able to interact with the pendulum system from the PC at an early stage of the design
- A circuit board (Xess XSA-200) containing the FPGA (Xilinx Spartan-II) and the required AD (the MAX122) and DA (the MAX530) converters, both from the MAXIM company

Regarding the software, the following tools will be used:

- MATLAB 7.0 with Simulink and the Fuzzy Logic and Real-Time Windows Target toolboxes (see [9] and [10]) for the initial design part of the lab
- Xilinx ISE WebPack 8.2i [11] for the part of VHDL coding and simulation and to transfer the final bit stream to the FPGA

IV. COURSE STRUCTURE

The proposed lab consists of the implementation of a fuzzy controller for the inverted pendulum using a FPGA. A block diagram of the target system can be seen in figure 3.

![Fig. 3. Block diagram of the target system](image)

In this section we continue describing the different parts of the lab and giving basic guidelines for implementing the specifications of each one.

A. PART 1: Background and first experimentations

The first part of the lab is devoted to review the basic principles of fuzzy logic and fuzzy logic controllers and apply them to stabilize the inverted pendulum system available in the lab. A PD (Proportional and Derivative) FLC with five triangle membership functions for its inputs and Takagi-Sugeno defuzzification (see [5]) scheme is suggested for the sake of implementation ease. In figure 4 it’s shown the suggested fuzzy system structure. Students’ first task will consist of determining proper missing values in the graphic axis.

![Fig. 4. Suggested configuration for the fuzzy controller](image)

Regarding the software, the following tools will be used:

- MATLAB 7.0 with Simulink and the Fuzzy Logic and Real-Time Windows Target toolboxes (see [9] and [10]) for the initial design part of the lab
- Xilinx ISE WebPack 8.2i [11] for the part of VHDL coding and simulation and to transfer the final bit stream to the FPGA

In figure 5 we can see the general architecture of a FLC. It basically consists of four parts

- fuzzifier
- set of rules
- motor inference
- defuzzifier

In our case we try to stabilize the inverted pendulum, so our reference is implicitly 0 rads, the input signal is the voltage to apply to the dc motor and the outputs are the angle of deviation from the vertical and cart displacement.
However, for implementing the PD FLC we don’t want the cart position but its speed, so the second input it’s indeed used to estimate the current cart speed. In figure 6 we can see the conceptual top level view of the system to design and implement.

The first task to be done consists of obtaining a set of rules for the fuzzy controller. This set of rules will be obtained using the Fuzzy Logic MATLAB toolbox and it will be tested via simulation using a model for the pendulum that will be given to the students. The next thing will be to test the set of rules directly against the real plant employing the Real-Time Windows Target toolbox connected through the PCI DAQ attached to the inverted pendulum system.

B. PART II: Design implementation

In this section we start the design of the digital system, the lab will follow a top-down decomposition methodology. The top level block is shown in figure 7. Data1 and data2 are the values coming from the angle and the position potentiometers through AD converters. Busy1 and busy2 are the ‘data-ready’ associated signals. Clk and rst are typical digital system signals. Turning to the outputs, data is the bus signal containing the digital output to be transferred to the DA converter. Ini-conv-DAC is used to start a new digital to analog conversion and clk6 is a clock needed by the DA converter. All the implementation will be based on VHDL coding, following as already commented a top-down methodology approach as to refine the design successively.

In figure 8 we can see a first proposed subdivision. The blocks are described in certain detail in the following sections.

Fig. 8. Architecture of system to be implemented

Fig. 9. Block diagram for U0 block

1) U0: INPUT SIGNAL ADAPTER: The purpose of this block is twofold, on the one hand it changes the AD converters two’s-complement format to an unsigned representation, on the other hand it computes the cart speed approximation from cart position readings.

Data1in is the word coming from the angle potentiometer digitized voltage and it’s directly connected to a register which changes the two’s-complement representation to an unsigned one. This register is the block on the right of the second row of internal blocks in figure 9. Data2in (cart position) follows a more involved path. First enters to one register, at the next instant of time the content of this register is transferred to another while a new data2 value enters to the previous one. These two registers are the inputs of a subtracter block which performs the difference. This difference then goes to an output register. The block in the middle of the second row in figure 9 is a control unit based on a state machine and is in charge of orchestrating the register to register movements through the proper switching of enable signals associated with the register blocks.

The output of the block are two positive values, one for the angle and the other for the cart speed (data1out and data2out, respectively).

2) U1: FUZZIFIER: U1, see figure 10, is mainly a combinational block in charge of fuzzifying the two outputs of the block U0 (angle and cart speed). In the rest of the discussion we make the assumption that a certain value for angle or cart speed can just belong to a maximum of two membership functions simultaneously. For each variable, the following triangular membership functions are considered:

- NH: Negative High
- NL: Negative Low
- Z: Zero
- PL: Positive Low
- PH: Positive High

The main inputs of the block are the adapted values of the angle and the cart speed (data1 and data2) and the AD converters conversion validations (busy1 and

Fig. 7. Architecture of system to be implemented

Fig. 10. Block diagram for U1 block
The outputs are four registers to indicate the degree of membership of each input to two membership functions (we supposed at the beginning, for the sake of simplicity, that in the worst case the maximum number of overlapping membership functions is two, see figure 11). The \textit{pert} output is connected to a 10-bit register (for our particular case) that codifies the degrees of membership functions as illustrated in figure 12.

The code is coherent with figure 11. In the first \textit{if} statement of the code \texttt{data1} belongs entirely to NH. So at the end of this \textit{if} we assign the 5 most significant bits of the register \texttt{pert} indicating that \texttt{data1} belongs just to NA (10000 pattern). The rest of the code treats exhaustively the rest of the remaining cases and deals with the typical sequential synchronization issues.

\section{U2: MOTOR INFERENCE}

U2 is the motor inference block of the fuzzy controller. This block receives as inputs:

\begin{itemize}
  \item \textit{Pert}, containing to what membership functions each of the two signal belong
  \item \textit{PertA1} and \textit{pertA2}, indicating the degree of membership of the angle respect to the membership functions indicated in \textit{pert} and \textit{pertB1} and \textit{pertB2} indicating the same information but referred to the cart speed value.
\end{itemize}

With this the U2 block has all the necessary information to calculate its outputs, which consist of five numbers indicating the height of each singleton membership function. The main part of this block consists of the implementation of a set of fuzzy inference rules as a set of \textit{if}-\textit{then} statements. See figure 4. Some sequential synchronization code is also needed.

\section{U3: DEFFUZIFIER}

In figure 14 we can see the block interface for the U3 block.

As it's known, the most implementation friendly version for the defuzzification block is the one based on the Takagi-Sugeno scheme. Basically, it amounts to implement the following formula:
\[ \text{control} = \frac{\sum_{i=1}^{n} w_i h_i}{\sum_{i=1}^{n} w_i} \quad (1) \]

where \( h_i \)'s come from U2 and are the heights of each output singleton membership function (outputs \( h_Z, h_{NH}, h_{NL}, h_{PH}, h_{PL} \), vertical axis in figure 15) and the \( w_i \)'s (relative weights, horizontal axis values in figure 15) are to be decided by the students and will be hard-coded inside the block U3, whose general structure can be seen in figure 16.

The result of applying equation 1 will enter the AD converter in order to apply to the dc motor the proper voltage. Both numerator and denominator can be easily computed in a purely combinational block. U30 performs this task. The implementation can avoid using multipliers if the weights \( w_i \) are power of two or sum of powers of two, as suggested in figure 15.

Blocks U31 to U35 implement the process of division needed in equation 1. The suggestion is to implement the traditional paper-and-pencil-method for division since it is simplified enormously when dealing with the binary case.

In order to implement the division procedure, we need:

- Two registers that contain the dividend and the quotient, blocks U32 and U35, respectively
- A comparer-subtracter block, U34
- Two auxiliar registers, U31 and U33

The process for performing the division is as follows. Once the numerator and denominator of equation 1 are calculated and are stable as U30 outputs, the first of this two outputs (the dividend) is loaded into the dividend register and the second one will be the second input of the subtracter block. Now the most significant bit of the dividend register is loaded into the partial dividend register (whose content is initially full with zeros) on its least significant position at the same time that its content is shifted to the left by one position. The subtracter block performs the difference between the partial dividend register (U33) and the divisor according to figure 16 connections schema, if the difference is positive then U34 outputs a ‘1’ to the quotient register. Otherwise a zero is output. If the bit quotient is zero then at the next clock cycle we load into the feedback register (U31) the result of the substraction operation, otherwise (bit quotient equal to one) it remains unchanged. Finally, the thirteen most significant bits of the feedback register are loaded into bits thirteen down to 1 of the partial dividend. The least significant bit (bit number zero) is taken from the most significant bit of the dividend register (U32) whose content shifts one position to the left again. This procedure is repeated until all the bits of the dividend register have been explored.

To illustrate better the procedure, in the following table we have the evolution of some of the registers for a particular case in which numbers of fourteen bits are divided:

<table>
<thead>
<tr>
<th>Registers</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>dividend</td>
<td>00101000000011</td>
</tr>
<tr>
<td>divisor</td>
<td>00000001111111</td>
</tr>
</tbody>
</table>

Fig. 13. U2 block interface

Fig. 14. Block U3 interface

Fig. 15. Singleton output membership functions (motor voltage) with possible example weights (1,64,128,192,255)

Fig. 16. General block diagram for U3 (the defuzzifier)
The synchronization between the different elements needed to perform the division is provided by a state machine (block U36) in charge of enabling at each time the proper registers. Besides, the state machine is in charge of changing the mode of operation of some registers which act as shift registers during the process of division and as standard registers before starting it.

Finally, block U37 adapts the signal output to put it in two's-complement and deals with the DA converter interface.

C. PART III: Simulation, synthesis and deployment

Once all the modules are coded, students will have to ensure their correctness, first individually, and then as a whole system. Obviously, simulations will be taking place as the students code new blocks, but for the sake of clarity in the presentation we have postponed talking about it until this section. The first simulations will be purely behavioral, logical simulations and post-route (or timing) simulations will be then performed before downloading the resulting design to the FPGA, see [11] for an overview of the simulation and synthesis processes using the Xilinx ISE software. Figure 17 summarizes the whole process associated to parts II and III of the lab.

\[
\begin{array}{|c|c|c|c|}
\hline
\text{clk cycle} & \text{partial dividend (U33)} & \text{partial dividend - divisor (U34)} & \text{bit to quotient (U35)} \\
\hline
1 & 00000000000000 & 1111110000001 & 0 \\
2 & 00000000000000 & 1111110000001 & 0 \\
3 & 00000000000000 & 1111110000001 & 0 \\
4 & 00000000000000 & 1111110000001 & 0 \\
5 & 00000000000000 & 1111110000001 & 0 \\
6 & 00000000000000 & 1111110000001 & 0 \\
7 & 00000000000000 & 1111110000001 & 0 \\
8 & 00000000000000 & 1111110000001 & 0 \\
9 & 00000000000000 & 1111110000001 & 0 \\
10 & 00000000000000 & 1111110000001 & 0 \\
11 & 00000000000000 & 1111110000001 & 0 \\
12 & 00000000000000 & 1111110000001 & 0 \\
13 & 00000000000000 & 1111110000001 & 0 \\
14 & 00000000000000 & 1111110000001 & 0 \\
\hline
\end{array}
\]

V. CONCLUSIONS

In this paper the contents of a new control-related lab course at the Autonomous University of Barcelona have been discussed. The implementation of a fuzzy controller is carried out using a FPGA as the target platform. VHDL is the chosen description language to specify the design. The authors really think the lab is going to be a good chance for students to improve their analysis and design skills when facing multidisciplinary problems, putting together knowledge acquired in separate subjects, i.e. control theory, intelligent systems and digital electronics. In addition, the gained experience in the use of FPGA technology is intended to contribute to the improvement of their curricula and be highly profitable for their engineering professional careers.

REFERENCES