A simulator of a reactor control system is presented. The hybrid simulator is based on a PC that contains the reactor dynamical model and a DSP TMS320C50 that executes the control algorithm. The control objective is to bring the reactor power from its source level (from mW to a few W) up to a full power level of 1 MW. The input signals to the DSP controller are the power error expressed in percentage and the reactor period that represents the time the reactor requires, at every sampling time, to increase the power by a factor e. The controller output that usually represents external reactivity has to be converted to the corresponding displacement of the reactor control rod. A reactor control rod driver prototype has been designed and built, and it is to be tested with the PS-DSP system.

KEYWORDS: Simulator, Control, Mechanical Prototype, Fuzzy controller, DSP.

1. INTRODUCTION

Some studies at the National Nuclear Research Institute of Mexico (ININ) aimed to introduce fuzzy logic based control techniques to regulate the power of ININ’s TRIGA Mark III research reactor. It is well known that fuzzy systems can formally incorporate heuristic knowledge and also deal efficiently with plant uncertainties and slow parameter variations. One of these studies [1] has shown the feasibility of using a Mamdani type fuzzy controller to bring the neutron power from 50W up to 1MW without violating at any time the reactor period limiting value. Another study [2] compares the performance of two methods to carry out the aggregation and defuzzification stages of a power controller. One of the methods discretizes the universe of discourse of the output variable whereas the other computes these stages in a continuous universe with closed form formulae. The comparison considered the accuracy and computational load of the controller, the required time to attain the desired power level, and the ability to maintain the reactor period always within safety limits. In another study [3] a reactor simulator was developed in Visual Basic, where the user defines the required final power and some of the plant and controller parameters were graphically and numerically available to the operator. The implantation of a fuzzy control algorithm in a DSP has also been done [4]. The present work extends the scope of these previous works. The objectives are: (a) Validation of the DSP results with respect to those obtained with a reference C++ program; (b) Integrate the DSP system with a PC that contains the point kinetic model of the reactor; and (c) Test the PC-DSP system together with reactor control rod driver prototype that has already been designed and built to this end.

2. PC-DSP-ROD DRIVE SYSTEM

A block diagram of the control system is shown in Fig. 1. The plant measured variables are the reactor period (T) and the neutron power (n). The power error in percentage (ne) is computed.
with respect to the power set point \( n_{\text{ref}} \). The values of \( T \) and \( n_e \) are sent to the DSP system, where the fuzzy control algorithm determines the slope of external reactivity \( m_{\text{ext}} \), which is then used to compute the amount of external reactivity \( \rho_{\text{ext}} \) that must be inserted into the reactor core. This amount of \( \rho_{\text{ext}} \) is converted to the corresponding displacement of the reactor control rod (a rod that contains neutron absorber material such as boron), thus increasing or decreasing the fission reactions (neutron power) inside the core.

**Figure 1. Block diagram of the hybrid simulation system.**

### 2.1 TRIGA Mark III Reactor Simulator

The dynamic equations that describe the reactor are known as the point kinetic equations, which are:

\[
\frac{d}{dt} n(t) = \left( \frac{\rho_{\text{ext}}(t) + \rho_{\text{int}}(t) - \beta}{\Lambda} \right) n(t) + \lambda C(t) \quad (1)
\]

\[
\frac{d}{dt} C(t) = \frac{\beta}{\Lambda} n(t) + \lambda C(t) \quad (2)
\]

\[
\frac{d}{dt} \rho_{\text{int}}(t) = -\alpha k[n(t) - n_e] - \gamma \rho_{\text{int}}(t) \quad (3)
\]

where: \( n \) is the neutron power, \( C \) is the concentration of delayed neutron precursors, \( \rho_{\text{int}} \) is the internal reactivity, and \( \rho_{\text{ext}} \) is the external reactivity. The rest are constant nuclear parameters. A higher-order Runge-Kutta method (RK) was used to solve the reactor equations [5]. The plant simulator was coded in C++. The main program has as input arguments the simulation time, the demanded power and the integration step. This program generates a .dat file with the values of time, power, and reactor period. This file can be used by another program (Matlab or Excel) with better plotting features. Fig. 2 shows the dialog window of the reactor simulator. After some validation tests of the C++ program, it was converted to Visual C++.

### 2.2 Fuzzy Control Algorithm Implanted on the DSP

The 2-input 1-output fuzzy controller consists of the following four stages: (a) Fuzzification; (b) Rule Evaluation; (c) Aggregation; and (d) Defuzzification. Each input variable is described with five fuzzy sets. Fuzzification of inputs \( n_e \) (power error in percentage) and \( T \) (reactor period) is carried out using the sets shown in Fig. 3 and Fig. 4, respectively.
The linguistic labels of these fuzzy sets are shown in Table 1.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Fuzzy set</th>
<th>Variable</th>
<th>Fuzzy set</th>
</tr>
</thead>
<tbody>
<tr>
<td>ne (power error in percentage)</td>
<td>GN</td>
<td>Negative Big</td>
<td>ne (reactor period)</td>
</tr>
<tr>
<td></td>
<td>MN</td>
<td>Negative Medium</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>PN</td>
<td>Negative Small</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ID</td>
<td>Ideal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PP</td>
<td>Positive Small</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Linguistic labels of the input fuzzy sets.

The fuzzy sets and their linguistic labels associated to the output $m_{pext}$ (slope of external reactivity) are shown in Fig. 5 and Table 2, respectively.

Table 2. Linguistic labels of the output fuzzy sets.

Table 3 shows the twenty five rules that relate the input fuzzy sets to the output fuzzy sets. A min-max inference mechanism determines the activation strength of each rule. The aggregation and defuzzification stages have been combined in one. The center of gravity (cog) method is used to defuzzified the aggregated output set. The crisp output obtained is the slope of the external reactivity from the current to the next sampling times.

Table 3. Rule evaluation.
An example of an arbitrary aggregated set is shown in Fig. 6. The universe of discourse of the output is discretized in such manner that the discretized cog is close to the exact cog.

The discrete formula used to determine the cog is

\[
\text{cog} = \sum_{i=-0.02}^{0.04} m_{pexti} \cdot \mu_A(m_{pexti})
\]

(4)

where \( m_{pexti} \) represents the i-th value of the output signal and \( \mu_A(m_{pexti}) \) the activation strength for that output signal value. The implantation of the control algorithm on the DSP is reported in [6]. However, two important aspects of the implantation are mentioned next. First, the fuzzy algorithm assigns any real value between zero and one as membership values. It also uses continuous universe of discourse for the input and output variables. To avoid the use of floating points numbers, a special format (defined as FPS=Floating Point Shift) was created that implies the scaling of the universe of both the input and the output variables. This scaling is made selectively, depending on the domain of each fuzzy set, resulting in integer number operations exclusively. Second, since the universe of discourse of the output \( m_{pext} \) takes values between -0.02 and 0.04, the DSP controller computes the inverse of the cog (computes the inverse of Eq. (4)), and sends the result to the PC, where the data is inverted to obtain finally the cog.

### 2.2 Communications Protocol

The fuzzy control algorithm is executed on the DSP and the reactor model on the PC (in Visual C++). A communication protocol is required between these two modules. The important issues concerning the protocol are explained in this section.

The architecture and the parallel port of the DSP are of 16 bits. This does not mean that the DSP cannot compute 32-bit operations [6], but only that the registers and memory are of 16 bits of length. On the other hand, the PC bus associated to the parallel port is of 16 bits and it is connected to the three registers used by the parallel port. However, even though the data register of the parallel port internally works with 16 bits, only 8 bits of the 16 go to the DB25 connector. This implies that 8 bits is the maximum data length in the communications protocol between the PC and the DSP. Eight bits of the control register of the parallel port could be used to augment the data length to be transferred, but the system would not have the possibility of implementing control signals to the hardware in which the data are sent or pass through. This situation leads to design a multiplexed data transfer, since the data received by the DSP are of integer type, and, by definition, an integer number in C++ is of 16 bits. Thus, the transfer will be done sending the 8 least significant bits first followed by the 8 most significant. It is worth mentioning that all the operations inside the plant simulator are of the floating point type. However, to send the input variables \( n_e \) and \( T \) to the DSP, there exists a function that rounds and converts their values to integer numbers. This function also divides the numbers, leaving them available to their sending through the parallel port. Thus, at least 4 data bytes will be sent to the DSP. Another task of the DSP related to the communications protocol is that of receiving the data and the subsequent concatenation to form again the 16 bit data. To this end, an action command language was created, which are interpreted by the DSP. This means that, when sending data, a command byte is first sent (see Fig. 7). The two least significant bits are used to define the DSP action. The actions are shown in Table 4. The remaining bits of the command byte are used to point out an address in case the command (01) requests the sending of data to the PC, thus being able to monitor the values of all the registers used by the control algorithm. The PC forms the command byte, divides the data to be transferred, and controls the transferring process to the DSP by means of control lines implemented with the control register of the parallel port.
Likewise, additional hardware is required for this communication process. The communication interface is composed of: (a) Latches in lock configuration; (b) An address decoder; and (c) A latch, which is exclusively used to send data to the power section that controls the positioning of the rod drivers. The schematic diagram of the communications interface is shown in Fig. 8.

2.3 Control Rod Driver Prototype
The block “control rod driver” in Fig. 8 is a prototype of the mechanism to position the control rods. This block is composed of a stepper motor driver, a power section, and the control rod mechanisms. A schematic diagram of this block is shown in Fig. 9.

The number of steps that a motor should move is received by an 8031 microcontroller that, in addition, generates the clock signal to indicate the motor speed. The drivers shown in Fig. 9 are the L297, which generate the pulse sequences of the stepper motors. These drivers limit the
current flowing through the motor windings. The power sections provide the required power to the motors. These power blocks consist of a L298N circuit, which is an array of power transistors. Finally, the stepper motors operate at 5V, 1.5A per phase with a resolution of 1.8º per full step. One of the control rod driver prototypes is shown in Fig. 10.

3. RESULTS AND CONCLUSIONS

With respect to the simulation of ascent of power, the integration step “h” influences the dynamic response. For instance, using h=0.005s the ascent of power presents slight oscillations. After several tests, the response with an h=0.001s did not show the oscillations produced by numerical approximations. The power response is shown in Fig. 11.

The control rod mechanisms are operating in open loop. The circuits to measure the position are being designed. They are important to validate that the real linear motion agrees with the motion demanded by the controller. Likewise, new functions are being aggregated to the simulator user interface, such as visualization bars or graphs to indicate rod position, power level, reactor period, etc. One of simulator windows is shown in Fig. 12.

4. ACKNOWLEDGEMENTS

This project received partial funding from DGEST under grant 503.05-P. It is also acknowledge and appreciated the collaboration of personnel of the Prototype Design Department of ININ in the design and construction of the rod driver mechanisms.

5. REFERENCES