Carlos Jesús Jiménez-Fernández
Dpto. Tecnología Electrónica
Universidad de Sevilla
Instituto Microelectrónica de Sevilla
IMSE-CNM CSIC/US
Sevilla, España
cjesus@imse-cnm.csic.es

Carmen Baena Oliva
Dpto. Tecnología Electrónica
Universidad de Sevilla
Instituto Microlectrónica de Sevilla
IMSE-CNM CSIC/US
Sevilla, España
baena@imse-cnm.csic.es

Pilar Parra Fernández
Dpto. Tecnología Electrónica
Universidad de Sevilla
Instituto Microlectrónica de Sevilla
IMSE-CNM CSIC/US
Sevilla, España
parra@imse-cnm.csic.es

Alejandro Gallardo Soto
Dpto. Tecnología Electrónica
Universidad de Sevilla
Sevilla, España
alegallardo@us.es

Francisco Eugenio Potestad Ordóñez
Instituto Microlectrónica de Sevilla
IMSE-CNM CSIC/US
Sevilla, España
potestad@imse-cnm.csic.es

Manuel Valencia Barrero
Dpto. Tecnología Electrónica
Universidad de Sevilla
Instituto Microlectrónica de Sevilla
IMSE-CNM CSIC/US
Sevilla, España
manolov@imse-cnm.csic.es

***Abstract*—The learning of digital design at the RT level by the students improves with practical work, which can be developed in teams, allow both the gradual advance of complexity as the learning progresses, and the proposal to be attractive to them, such as playing simple games. FPGAs and development boards offer a very suitable platform for the implementation of these designs. This paper presents a work in the Advanced Digital Design course (4th year of the Degree) consisting of the construction of a slightly adapted version of the game "Simon Says" in which the player must memorize a sequence that becomes more difficult for as levels pass. The work, which occupies the second half of the semester, is carried out by teams of three students and must have a demonstrator implemented on a Digilent Nexys4-DDR board.**

Keywords— VHDL, design of digital systems, games in VHDL, FPGA.

I. INTRODUCCIÓN

The subjects of digital design in the last degree courses in electronic degrees allow the use of FPGA device and hardware description languages as a platform to experimentally test the designed circuits. This methodology has many advantages. One of them is the use of a single CAD environment for the design, verification and programming of the devices. In addition, the software is offered free of charge for educational use by the manufacturers of FPGA. Another advantage is the availability of development boards, which include, in addition to the FPGA, display and interface elements that allow input values to be input and output results to be viewed. With all this, teaching of digital design with the VHDL-FPGA tandem is an alternative with an acceptable cost and, above all, very practical and highly attractive to students.

In the Advanced Digital Design course of the fourth year of the Degree in Industrial Electronic Engineering, taught at the Higher Polytechnic School of the University of Seville, the student is intended to learn the most important concepts of digital design. It is taught how to describe digital circuits using the VHDL hardware description language and how to implement them on FPGA devices (in our case from Xilinx).

The student has to carry out a proposed project on a Digilent Nexys4-DDR development board available in the laboratory.

In this communication we present a proposal carried out as teamwork within the subject. It is about the development of a game of skills and memory similar to the well-known "Simon Says". Since the design is quite complex, it is carried out as a second work of the subject and it is developed by teams of three students. In each team a distribution of tasks is proposed between the components both at design and verification level, which will train communication, leadership and collaborative work skills.

This work involves learning multiple concepts since, in addition to the VHDL designs themselves, it is necessary to use an additional module consisting of an LED matrix that has to be connected to the development board to display the sequential images that are part of the game.

The structure of this communication is as follows: the second section explains the mechanism of the game. Section III details the design to be carried out by the students. In section IV, the implementation and the experimental results are presented. Finally, the conclusions are summarized.

II. GAME SPECIFICATIONS

The task at work, as previously mentioned, is to emulate the well-known game "Simon Says".

The goal of this game is to remember in order a sequence of events that occur and be able to reproduce it. The length of this sequence of events is progressively increased as you advance in the game. In our case, the events to memorize are five different drawings, specifically: a circle and four arrows: up, down, right and left that will be displayed in a 8x8 led matrix. This matrix is inserted in a board ("Fig. 1") that is the result of a previous work and it is connected to the Nexys4 through the PMODS ports.

The images displayed on the LED matrix are generated in a (pseudo)random way. It starts with a single image that must be remembered and returned by the player. In the next step, which involves going up a level, a new image is added and displayed in sequence with the previous one and so on, with one image added each time a level is passed. The number of images in the sequence represents the level reached, which is displayed on the Nexys4 7-segment displays. In case of failure, a new game is started with a new and different sequence.

Each image must be visible for a certain reasonable time, then it is turned off, the next event is shown, it is turned off again and so on for all the drawings of that level. Once all the images are shown, the player will be expected to repeat the sequence.

To replay the sequence, the player must press one of the five buttons on the Nexys4 board (one per event). The five buttons are arranged on the board as a cross, which makes it easier to associate each of the buttons at the edges with the direction of each arrow and the central button with the circle. If the sequence is correct, it goes to the next level, but if it is incorrect, an X must be shown on the LED matrix and the game is over.

The status of the game is indicated by three colors of a colored LED included in the Nexys4. A blue led will light up when it can be pressed, green when it is pressed correctly and red when it has failed.

In the following section we will describe the system to be developed.

III. System Design

Below are the most significant details of the system that the team of students has to design. The main module called "simon_says" ("Fig. 2") consists of the following inputs and outputs.

Starting with the inputs, we have:

- clk: clock signal, the Nexys4 board clock, which is 100 MHz.
- reset: Asynchronous reset signal active on high. It will be connected to the switch SW0 on the board.
- btnu, btnd, btnr, btnl y btnc: which will be connected to the buttons on the board.

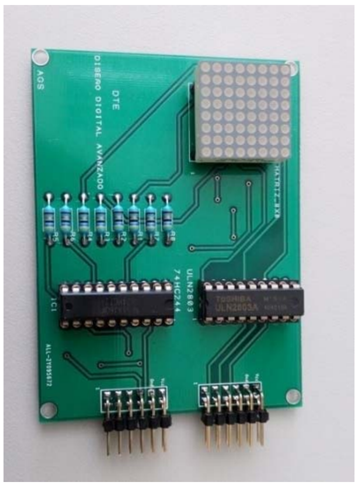

Fig. 1. Board containing 8x8 LED matrix

- start_game: It will be connected to the switch SW1 on the board.

Regarding the outputs:

- row: 8 bits indicating which row of the LED matrix will illuminate.
- col: 8 bits indicating which LEDs will light for a particular row of the LED array.
- led_r, led_g: signals to set the tricolor led to red or green depending on the game status.
- an: 8 bits that enable the activation of each of the 7-segment displays on the board.
- ca, cb, cc, cd, ce, cf, cg: signals that control the segments of each display.

It is recommended that the design has a hierarchical structure, with the following modules: rom_simon, ram_data, lfsr_16, debounce_cod, bin2bcd, bcd7seg ("Fig. 3").

The rom_simon module consists of a ROM that will store the encoding of the images to be displayed on the 8x8 LED array. Each symbol requires 8 rows of 8 bits. At least seven images must be stored: the five that can be part of the game sequence, the X to indicate failure and another to turn off the whole array. Therefore, a 64 word 8-bit ROM with addr_row address bus and data_row data bus will be generated. The seven images referred to are shown in "Fig. 4". To generate the content of the ROM, an LED matrix editor can be used.

The ram_data module is a RAM memory with double port, one for writing and another for reading. In this memory the pseudo-random sequence of drawings to be remembered will be stored. The maximum number of values to remember will be 128 and as there are 5 possible values, 3 bits are needed to code them (codes from 0 to 4). Therefore, this memory is a 128-word RAM of 3 bits with addra and addrb address buses and dia data buses for writing and doa, dob for reading.

The sequence of images to remember is generated in a pseudo-random way and it is stored in the ram_data module. In order to generate this sequence, we are going to use the lfsr_16 module. It is a shift register with linear feedback: with the reset an initial value is loaded and in each clock cycle a shift is made from the less significant bit to the more

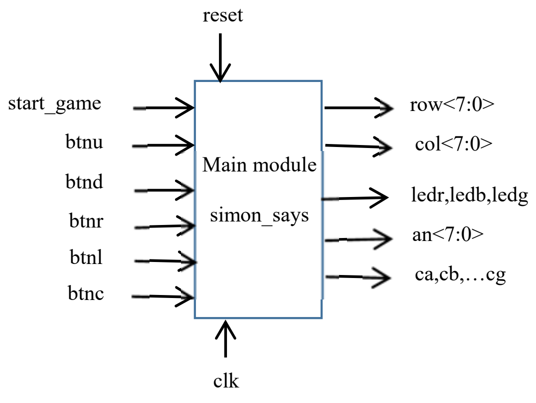

Fig. 2. Block diagram of module simon_says.

significant bit. The least significant bit is loaded with the XOR operation of the several bits register itself. This design introduces the XOR operation of the bits 10, 12, 13 and 15. The initial value to be loaded when resetting cannot be all the bits at zero.

This module has as inputs the clock and the reset and as output the 16 bits of the shift register of which, the three most significant ones, are connected to the input data bus (dia) of the RAM for writing.

For the system to work properly it is essential to eliminate the bouncing that occurs when a button is pressed and released because otherwise the circuit will interpret it as if it had been pressed twice and therefore detect an incorrect sequence. Five bounce filter circuits are used, one for each button, named debounce_u, debounce_d, debounce_r, debounce_l and debounce_c. The students can choose the specific form of implementation but must explain how they have designed it.

The debounce_cod module has two functions: first, it provides a split clock signal to the debounce modules, so that they consume less resources. Secondly, it provides the simon_dice design with a 3-bit signal where the button pressed is coded (5 codes) and a different code that indicates if more than one button has been pressed at the same time. Please note that the code must match the one stored in the ram_data and the display in the 8x8 LED matrix.

Finally, code converters are used to display on the Nexys4 7-segment displays the level of play as you progress.

The entire system is written in VHDL code. This includes, on one hand, the description of the modules already presented, and on the other hand, the description of the main module simon_says. In this last one, the connection between the modules of the hierarchy is made and the definition of several processes is included. These are, the generation of the pseudo-random sequence, the writing of the RAM with the random sequence, and the control of the game: start of the image display process, capture of the player's clicks, check of coincidences, storage of the achieved level, level advance or failure detection and, in short, all the necessary tasks for the game operation.

As for the generation of the pseudo-random sequence it is recommended, to simplify the process, that every time a game is started the pseudo-random data of the 128 levels are generated and written in the ram_data. To do this, in a process the 3 most significant bits of the output of the component lfsr_16 are taken. If the value is between 0 and 4, it writes it into memory. Otherwise, it lets 8 clock cycles pass and takes another data. When the 128 values have been written, it will set a signal to '1' indicating that the different phases of the game can begin.

It is recommended that all phases of the game be controlled in one process. This process will have to do the following tasks:

- The process will have to be started when no game is being played and the start_game signal is set to '1'.
- Once started, the sequence to be remembered must be displayed on the 8x8 LED matrix. To do this, it will keep the corresponding image for a certain time, then it will keep the matrix off for a certain time and then it will show the next image, and so on until it has shown all the images corresponding to the level in which the player is.
- Then check the sequence of buttons pressed. This phase starts with the lighting up of the blue led. Then, every time it is correctly pressed, the green led lights up.

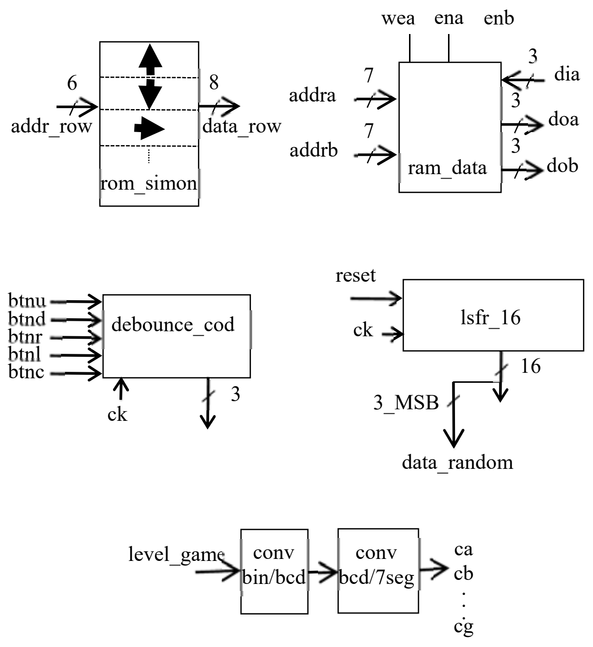

Fig. 3. Most significant internal blocks of the main module.

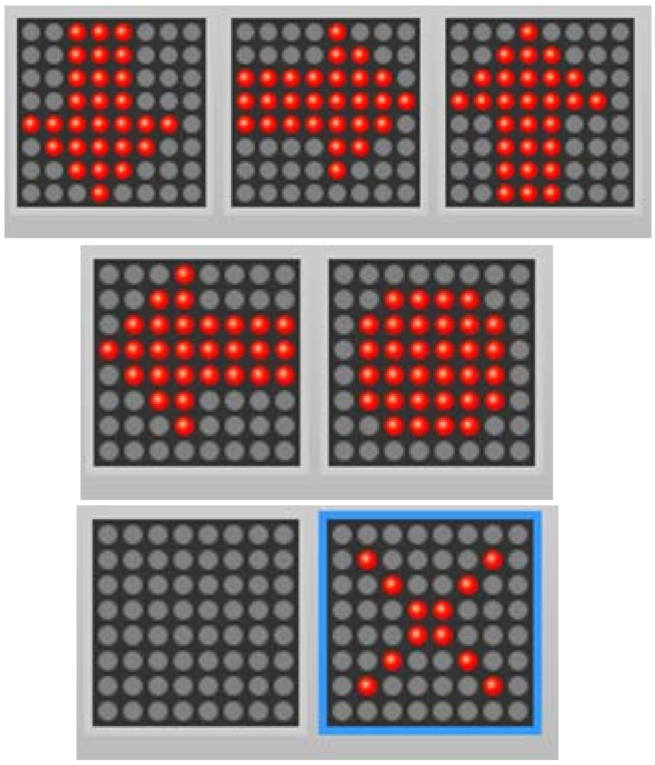

Fig. 4. Images of the game on 8x8 LED matrix.

- If the sequence is correct, the level is increased and the same sequence as above is displayed by adding one more.
- If the sequence is incorrect, the matrix shows the fault indication (X) and the red LED lights up.
- Finally, wait until the start_game entry is set to '1' to generate a new pseudo-random sequence and start again.

To display the images on the 8x8 LED matrix, it is recommended that this is done by reading the rom_data memory directly. To do this, the process that controls the operation of the game sets the 3 most significant bits of the address bus of the rom_data memory (information read from the ram_data memory), to indicate the image to be displayed in the array. The three less significant bits are changed sequentially and used to decode the array row (row output). At each moment, the value of the column of the array (output col) will be the value stored in the corresponding address of the memory rom_datos.

For example, assuming that the left arrow starts in the lowest directions in memory, we would have its 8 rows in the binary directions "000_000" to "000_111". If the right arrow is the following its directions would be "001_000" to "001_111", and so on. Therefore, if the left arrow is to be displayed, the process controlling the game phase would set the three most significant bits ("000") and another process would cyclically generate the three least significant bits ("000" to "111"), so the values would be read from the ROM and displayed in the matrix.

The following section details how the work has been implemented within the subject.

IV. Implementation and Results

This work is developed as a team task, to be developed as part of the non-presential credits.

The team, formed by three students, receives the necessary documentation that includes: objectives of the work, general description of the game, description of the design to be made with indications of how to carry it out and instructions for the verification of the system. They are also encouraged to make variations of the game and add it as a complementary design.

Students have one month to complete the work, during which they can attend tutorials.

The design environment they should use is ISE Design Suite 14.7 from Xilinx. To use it, they can access the course lab where, in addition, they are provided with the development boards and the board with the LED matrix.

As a result of the work they must deliver:

- A zip file containing the design of the complete project. This must contain the design files and all the test_bench files used, also the pin restriction file. In addition, they must deliver the ".bit" board programming file.
- A document that details in a clear and concise way how each of the designs has been made, detailing the strategy followed to make the debounce circuit and explaining how the process of the game phases has been carried out.
- A table with the work done by each student.
- A list of the most significant difficulties found and how they have been resolved.

Once the work has been finished, all teams must hold meetings with the teacher to defend it. The evaluation takes into account the contribution of each student to the project and, although it is a team work, the grade is individual.

Below are some photos obtained after the implementation, where you can see the system in operation at different stages of the game.

Specifically, "Fig. 5" shows the board with the FPGA already programmed at the moment immediately before the start of the game.

In the "Fig. 6-8" you can see the board in three different levels of play and with the matrix showing some of the symbols.

Finally, "Fig. 9" shows the board in the player's fault situation, which is identified by the red diode on and the cross shown in LED matrix.

V. Conclusions

With this work proposal, the student is confronted with the development of a sufficiently complex design based on some very general indications. In addition, the student carries out the complete process of design, that is to say, the writing of the VHDL code, debugging of errors, writing of the stimulus file (testbench) that allows the functional simulation, the implementation in the board and its experimental check.

Emphasis is placed on teamwork, which includes the ability to establish an appropriate distribution of tasks and roles, as well as the development of oral communication skills, both among classmates during the development of the activity and with the teacher when presenting results.

As another conclusion, it can be added that the students successfully carry out the work, even proposing specific modifications or extensions in some cases.

Acknowledgements

This work has been partially supported by the project INTERVALO (TEC2016-80549-R).

References

[1] The teaching project can be consulted at the link: https://www.us.es/estudiar/que-estudiar/oferta-de-grados/grado-en-ingenieria-electronica-industrial/2010034 (accessed in March 2020).

[2] LED matrix editor: https://xantorohara.github.io/led-matrix-editor/

[3] The Nexys4 development board specifications can be found at: https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/reference-manual

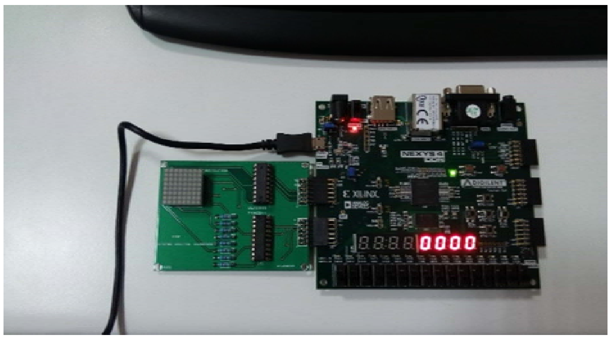

Fig. 5. Board before the start of the game.

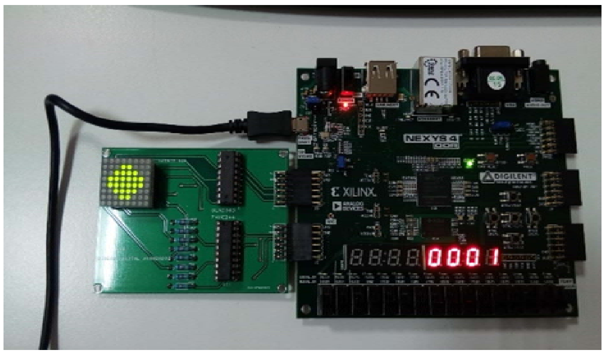

Fig. 6. Board at level 1 game.

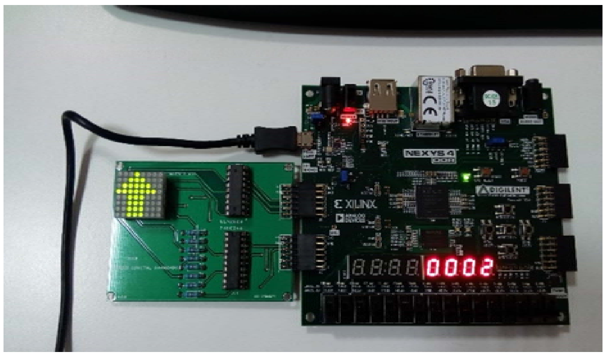

Fig. 7. Board at level 2 game.

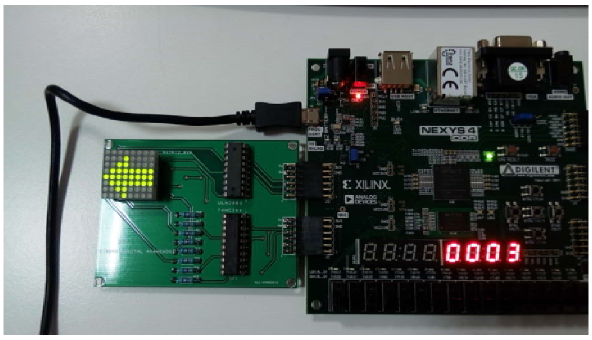

Fig. 8. Placa en el nivel 3 del juego.

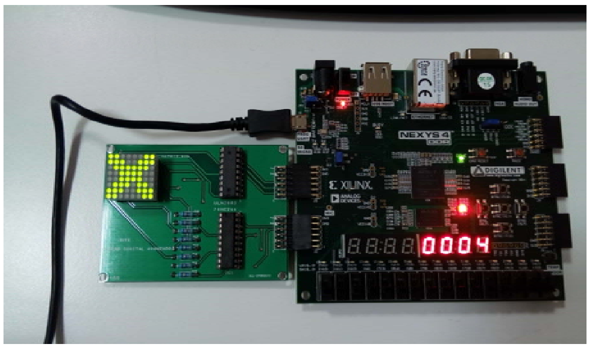

Fig. 9. Placa en el instante de fallo del jugador.