CS223 Project Assignment

In the project you are expected to learn how to use the 7 segment module, how to give input via switches, how to use the buttons and design cellular automata for a 8x8 matrix on the BetiBoard.

Project has 3 parts:

- Part 1: Putting values into the 7 segment module via switches. This part can be completed on your Basys boards.

- Part 2: Saving values displayed on 7 segment into the memory and show on LEDs. This part can be completed on your Basys boards.

- Part 3: Implementing a game with its logic. This part requires integrating your Basys implementation with the Beti board.

Part 1:

In this part you should write a module, which assigns values into the 7 segment display. Values should be taken from switches. You should assign each value one by one. You should use 4 switches to get the inputs ranging from 0 to 15 (HEX digit). So you should convert a binary value (given on the switches) to its hexadecimal counterpart (value displayed on the 7 segment display). You can assign four values via buttons or switches (choice is yours).

Option 1 - using buttons: Active digit (which value will be assigned into) should blink. When digit is determined, button should be pushed. After button is pushed, next digit will be active. Proccess will be finished with pushing the button when last digit is blinking.

Option 2 - using switches: Active digit will be selected by 2 switches which provides 4 differnt combinations. To assign the value, you should use another switch. When this switch is on, value will be assigned to the according digit. When it is off, no action is required.

Part 2:

In this part you should write a module, which stores values in the range of 0 to 2^{64}-1 . This module should be integrated with the first module, and it should take values from 7 segment module four times. This procedure may be done similar to the first part either with buttons or switches.

Also this module needs to show stored values on the LEDs. The value should be partially shown in binary form since there are 64 bits and there are 16 LEDs. To show all digits of the value, 2 switches should be used. 2 switches provide 4 different cases, hence 16x4=64 bits. When user presses the start button, this stage ends and the game starts, which is described in part 3.

Part 3:

In this part, you should provide sequential logic for cellular automata (explanation below) with the following rules, which will be selected according to your Bilkent ID number. Stored data will change 1 step when button is pressed. You will use 4 different buttons for different cell groups. You should count the number of button presses which will show as the score in the seven segment module. When all cells are off, the game is over. After this, the final score should be blinking with 0.5 second period. So, the seven segment should be enabled for 0.25 second and disabled for 0.25 second continously. If user pushes the reset button, game state should go back to part 1. If user pushes restart button, game should restart with the same initial state. Reset and restart functions can be activated any time during or at the end of the game (i.e any time during Part3 is active). Make sure that your Part 3 module works with the module you implemented in Part 2. You will be provided an interface which converts the data to 8x8 LED matrix output.

A. What is cellular automata ?

Cellular automata is a system of finite state machines. It consist of regular grid of cells. Each cell is in on or off state determined according to the neighboring cells. You will use horizontal and vertical neighbors of the cell to determine the next state of the cell (North, South, East and West).

Example Rule 1: A cell turns on if all neigbors of a cell are on.

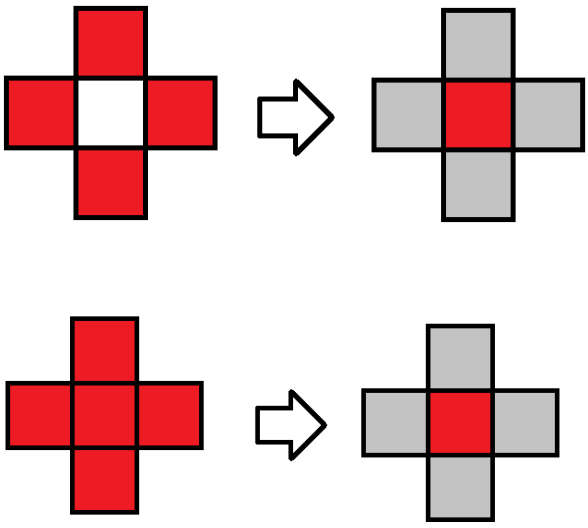


Figure 1: In both cases, center cell is turned on since all neigbors are on. (Red indicates on state, white indicates off state and gray is ingored)

Example Rule 2: A cell turns off if vertical neigbors are on and horizontal neigbors are off.

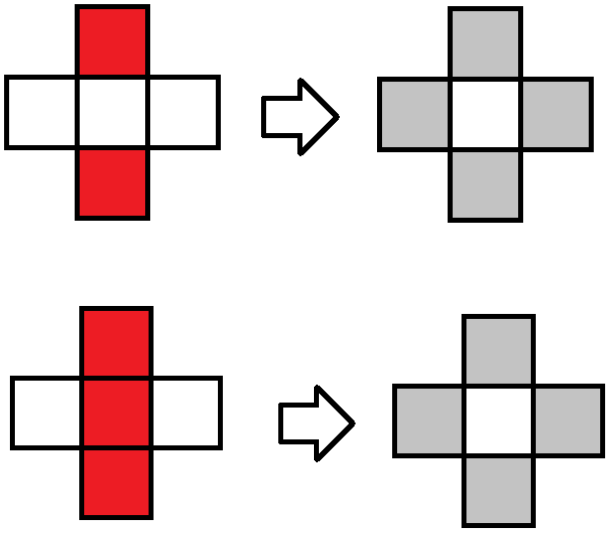
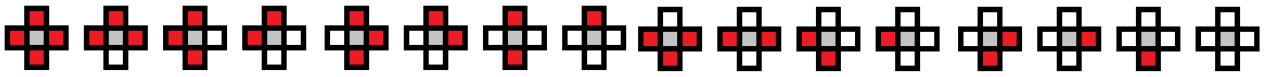


Figure 2: In both cases, center cell is turned off since vertical neigbors are on. (Red indicates on state, white indicates off state and gray is ingored)

B. Which rules to apply?

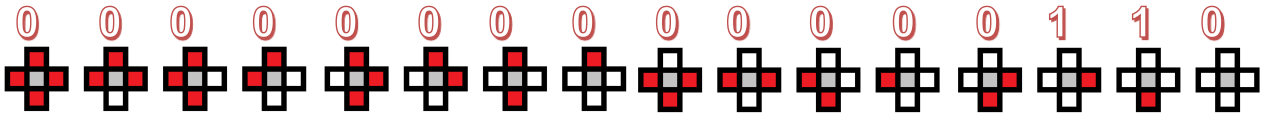
Note that, this rule selection should be done with pen and paper, not programmatically. Rules are indexed in a range from 0 to 2^{16}-1. To find the behavior of the rule, index should be written in binary form. If a digit is 1 corresponding state is on, otherwise 0. Digit – state relationship as following:



most significant digit least significant digit

Figure 3: Digit vs. state transition is according to the bit representation.

Example: For Rule 6 (binary representation is 0000000000000110), a cell will be on next state only if SOUTH and EAST neighbors are on and the others are off. That is, you select the bits that are equal to 1 in the binary representation and AND them. In the example figure below, binary value above the neighbour state depicts if the value of the middle cell is ON or OFF.



most significant digit least significant digit

Figure 4: Example mapping of Rule 6 to state-transition.

C. Rule index calculation from ID

Index will be calculated as following:

Rule Index = (ID % 32768) * 2

To determine next state of all cells, you use this rule according to your student ID.

Example: Let student ID be 21500000

Rule Index = (21500000 % 32768) * 2 = 4192 * 2 = 8384 (in binary form 0010000011000000). Your rule that you will apply during the game will be based on this binary representation. Again, note that these calculations and selection of rules should be done on pen and paper.

D. Some specifications for corner cases

Since 8x8 matrix is finite, we have edge and corner cells. For the first row, you are expected to use the last row as NORTH (top) neighbor. For the first column, you should use the last column as the WEST (left) neighbor. Similarly, SOUTH (bottom) neighbor of the last row is first row, and EAST (right) neighbor of the last column is the first column.

Example:

The center cell(C) and neighbors, i.e., top (T), right(R), left(L), and bottom (B) are given in the figure with respective colors.

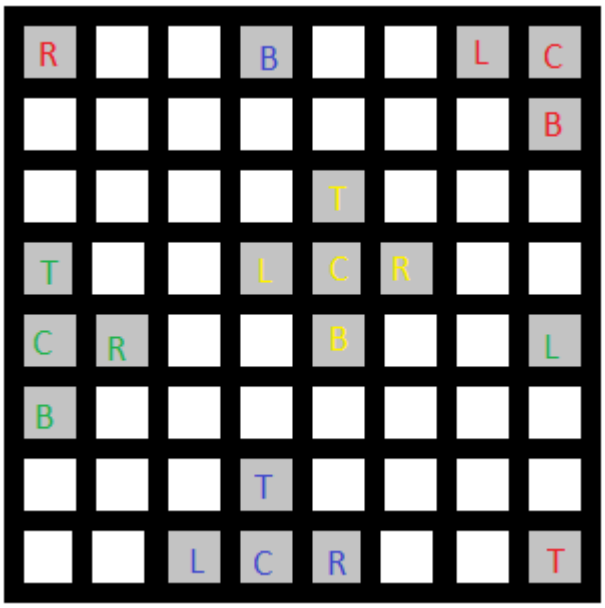


Figure 5: Center (C), top (T), right(R), left(L), and bottom (B) cells.

E. Cell groups

Cell groups are used in order to choose how to trigger the button for each cell. There will be 4 trigger buttons and 4 cell groups respectively. Each trigger button will be connected to 16 cells. When button is pressed, these 16 cells will move on the next state according to the rule, the other cells won't change. You can assume that only one button will be pushed at a time (No two buttons pressed simultaneously). For each push, you should increment the score on the seven segment by 1. Cell groups are determined by the **last four digits** of your student ID. Each digit determines one of the 4 quarters as following:

Example: Let's assume your ID is 21X0**ABCD**

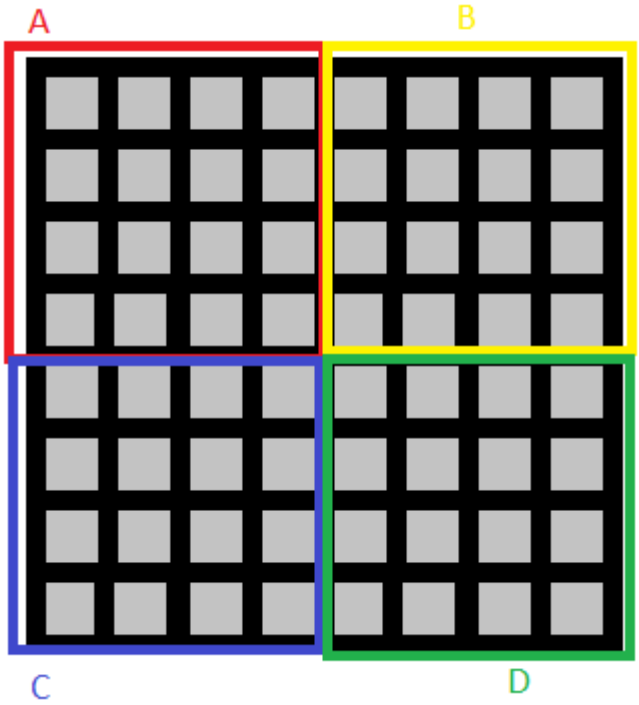


Figure 6: Cell groups are determined by the last four digits.

For each quarter, you determine the groups of cells. For each respective digit, calculate <Digit> modulo 4. According to the result, label the cells as following:

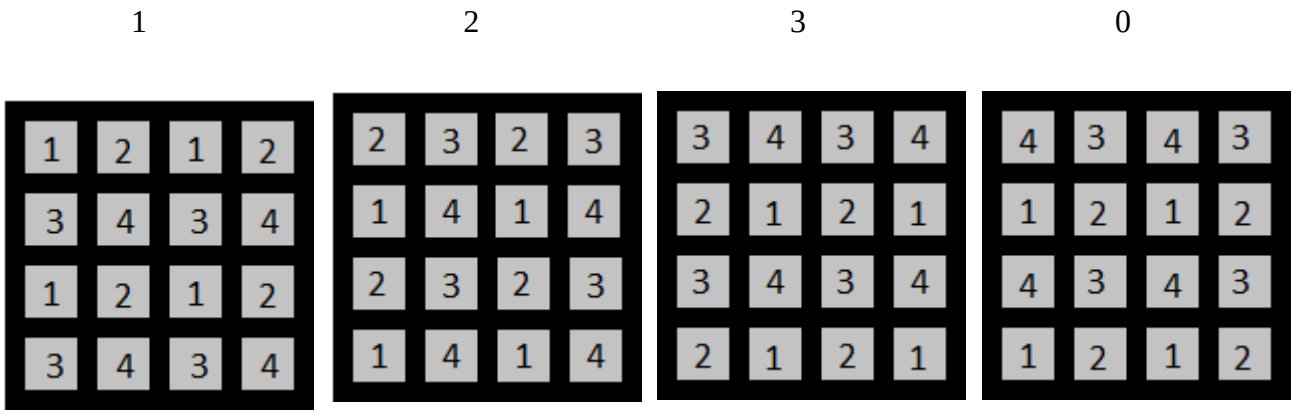


Figure 7: Cell group labels.

Cells with the same label are considered to be in the same group. For example each cell labeled with 1 will be activated to apply the rule simultaneously when button 1 is pushed. You should use four buttons on the Basys-3 for all 4 different groups.

Example: Assume 21001234 is the student ID number, take the last 4 digits of the ID, that is 1234. Then, for group A 1 is used, for group B 2 is used, for group C 3 is used, and for group D 0 is used. Based on these values, 8x8 will be grouped as following:

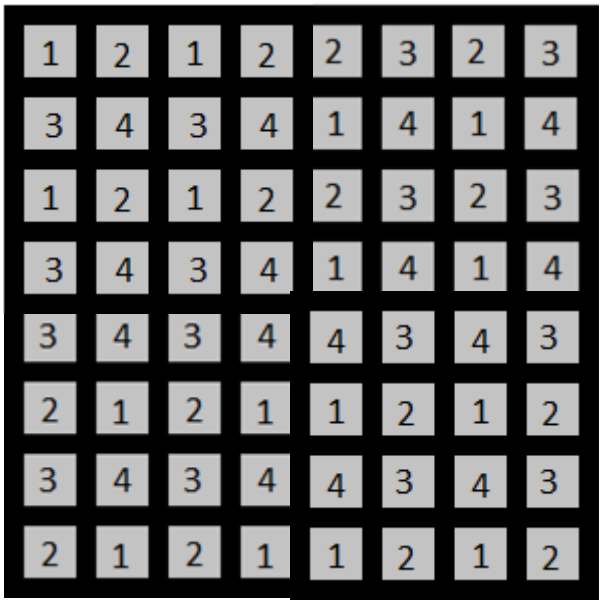


Figure 8: Example cell groups for ID 21001234.

F. 8x8 Matrix interface

You will be provided the following module which converts your data into the 8x8 matrix:

```
module converter(     input logic clk, input logic [7:0][7:0] data_in,
                      output logic[7:0] rowsOut,
                      output logic shcp, stcp, mr, oe, ds );
```