MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

PS2 Keyboard

PROIECT DSD

Student: Borz Robert-Ionuț
Cadar Ștefan

Coordonator: Blaj Ileana

2022

DSD Project: PS2 Keyboard

Project task

Implement a **PS2 keyboard controller**. It must read from a keyboard and display the characters on the 7-segment displays . Only 8 characters will be displayed on the board and certain keys will have special roles. The enter will start a new line by resetting the displays. The position of the dot will be changed with the arrows. The backspace will delete the last character. If the dot is not on the last position the backspace deletes the character before the dot. This project will be done by 2 students.

The main components of the project

The black box of the project can be observed in Figure 1. The inputs are “clk” which is the clock of the FPGA board, “ps2_clk” which is the clock of the keyboard and “ps2_data” which represents the data received from the input device. The outputs of the design are taken by the FPGA and used to display the characters on the board.

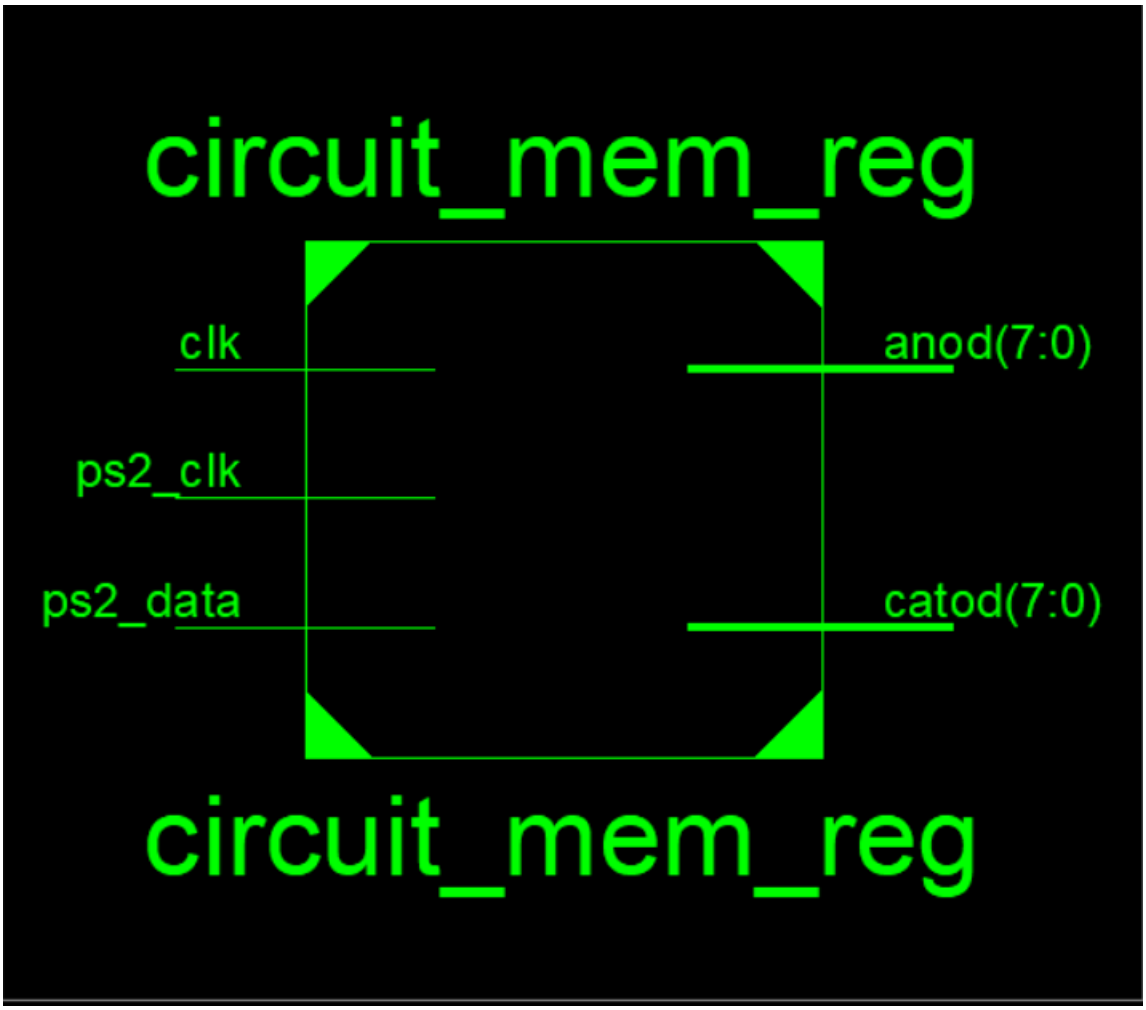

Figure 1

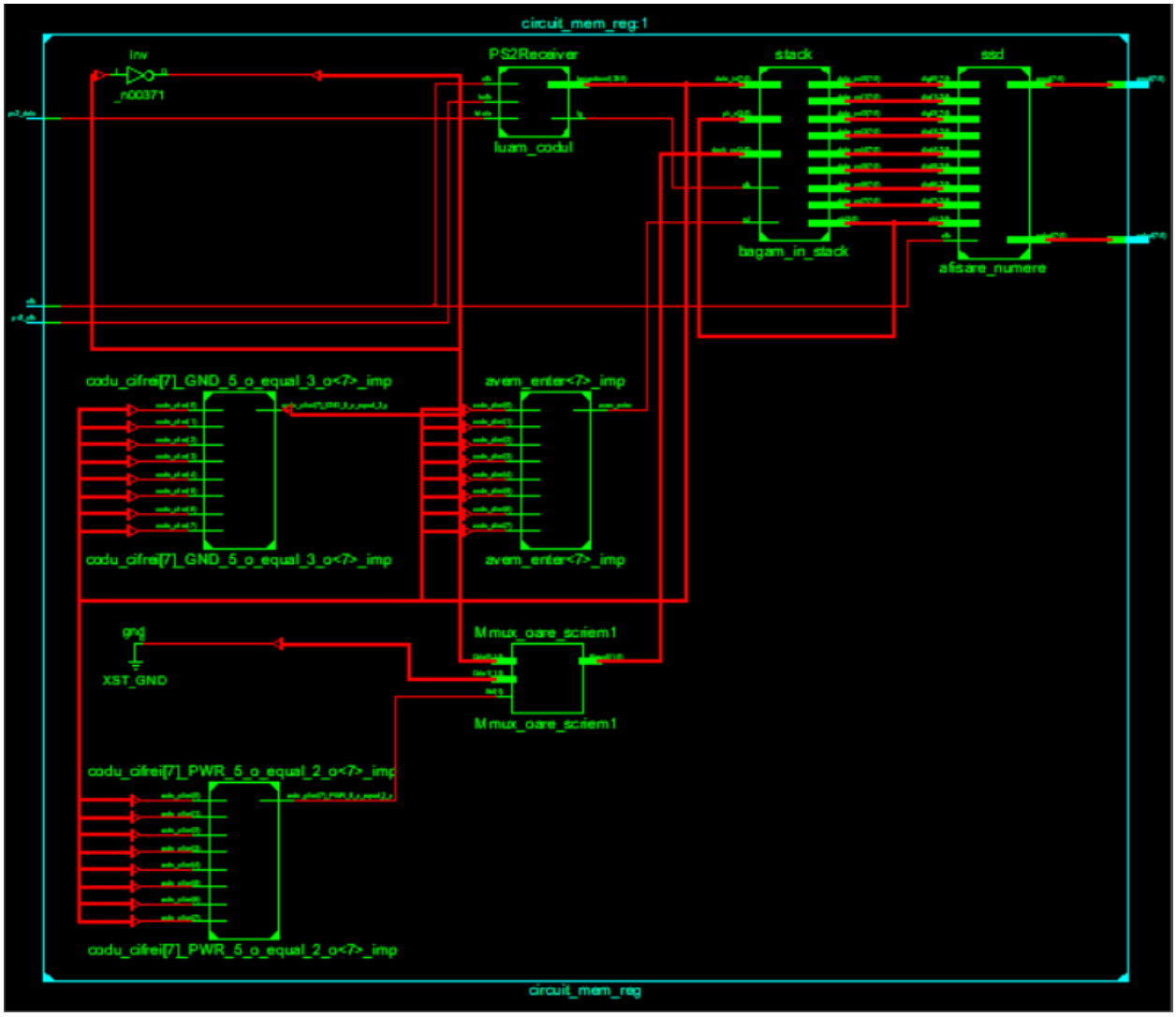
circuit_mem_reg:1
inv
_n00371
PS2Receiver
luam_codul
stack
ssd
bagam_in_stack
afisare_numere
codu_cifrei[7]_GND_5_o_equal_3_o<7>_imp
codu_cifrei[7]_GND_5_o_equal_3_o<7>_imp
avem_enter<7>_imp
avem_enter<7>_imp
gnd
XST_GND
Mmux_oare_sciem1
Mmux_oare_sciem1
codu_cifrei[7]_PWR_5_o_equal_2_o<7>_imp
codu_cifrei[7]_PWR_5_o_equal_2_o<7>_imp
circuit_mem_reg

Figure 2

Design and implementation

Inside the design we have 7 main components. First is the ps2 receiver which takes the inputs from the keyboard and puts them in a signal which contains the last 3 characters written. Then we have 3 comparators for the last character written. If the code is “F0”, this means that the key has been lifted, this will be ignored and our design will be on hold. The other two comparators are testing for the special keys enter with code “5A” and backspace with its code “66”. With a multiplexor we chose depending on the key read, which function of our main memory component we will use. This component is a stack which can push, pop, reset and hold. We decided to use a stack for our implementation, because it suits our needs. The way a text editor writes characters is similar to how a stack pushes an item. The backspace is actually just a pop. Furthermore, the stack also has a pointer which represents on which 7-segments display will have the point. This is also useful when we use the arrows. Writing at a certain position will overwrite the current character displayed whereas popping will also initialize a shift just like in a usual text editor. The last component is a SSD, which also converts the 8 bit code form the keyboard and converts it into codes for the 7 segment. It can print all letters in the english alphabet and every decimal digit. This component also decides on which of the 8 positions to put the dot. All the components can be observed in Figure 2.

Functioning and Use

The point will always start at the left most segment. We must first press any key to initialize the project after every enter. In the initial state of the FPGA board will look like in Figure 3.

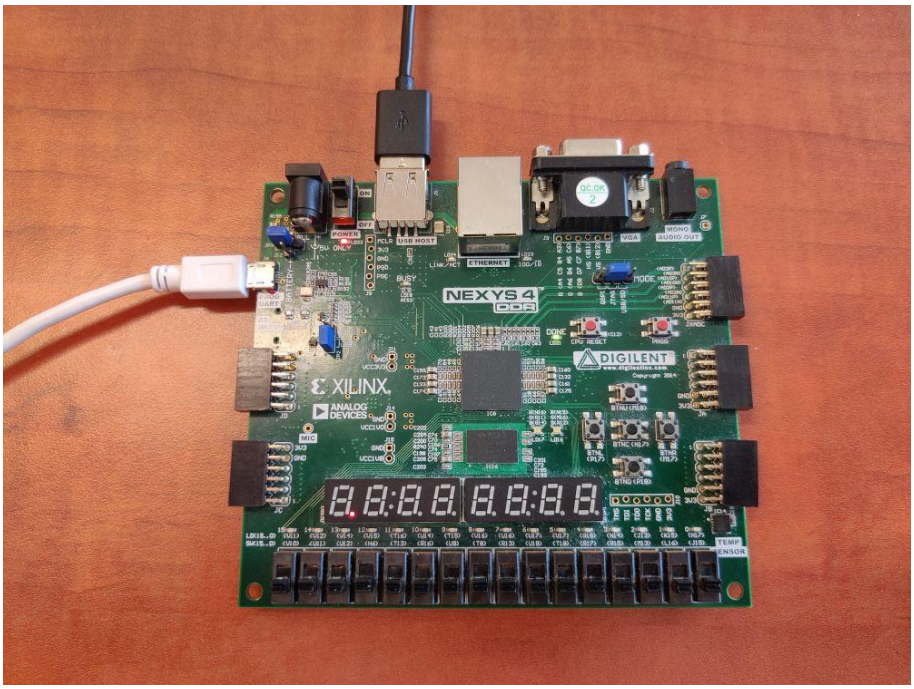

Figure 3

After pressing a key the character will be displayed on the segments and the dot will be shifted, as we can see in Figure 4.

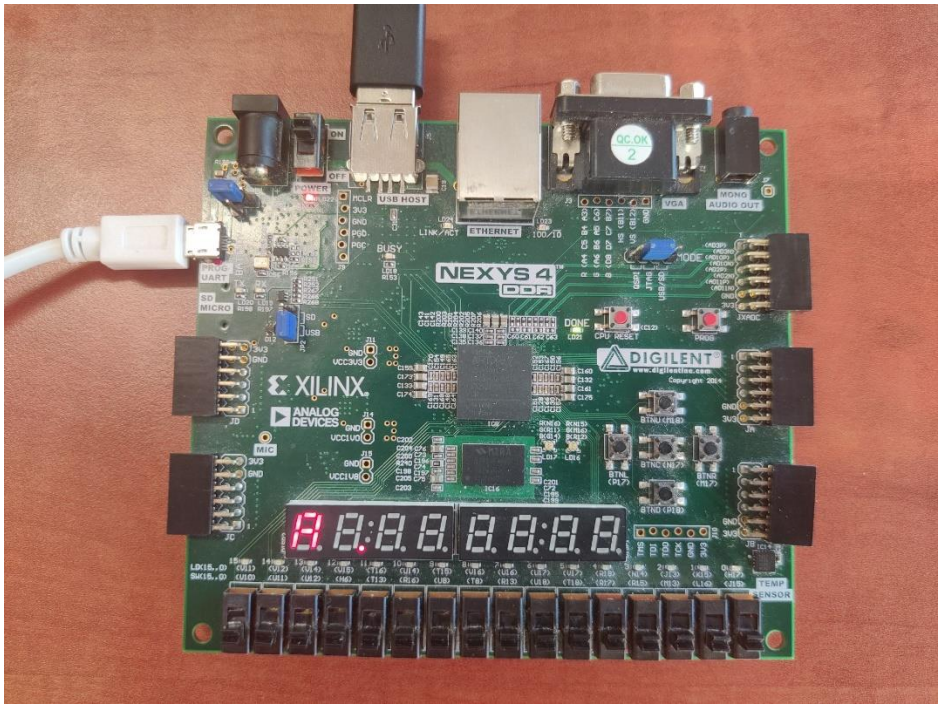

Figure 4

After pressing enough keys to fill all the segments an additional pressed key will be placed on the last position and the rest of the keys will be shifted to make room for other characters(Figure 5 and Figure 6).

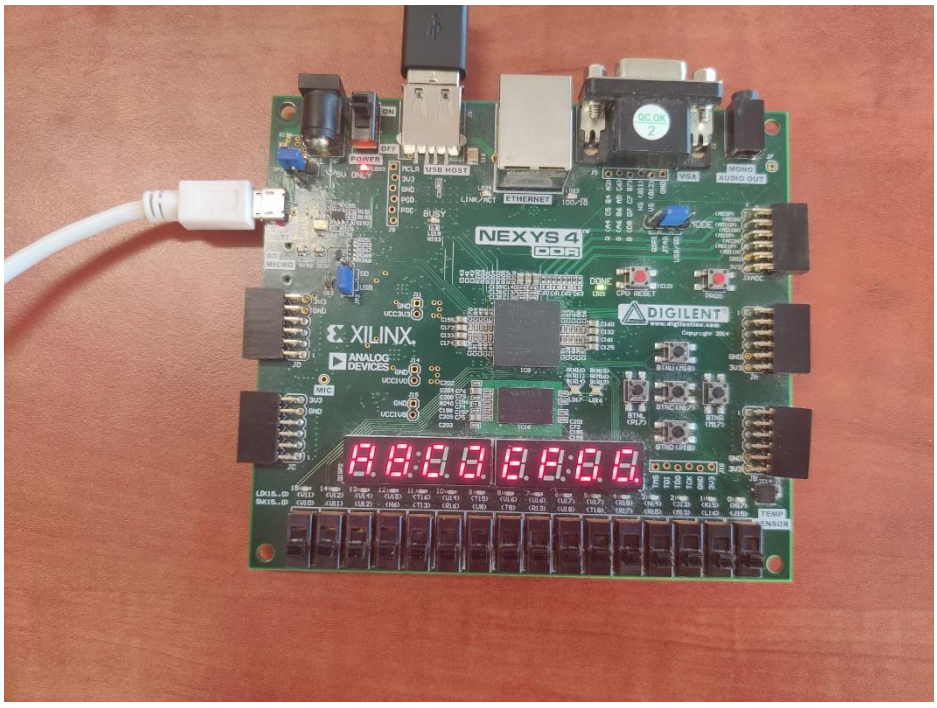

Figure 5

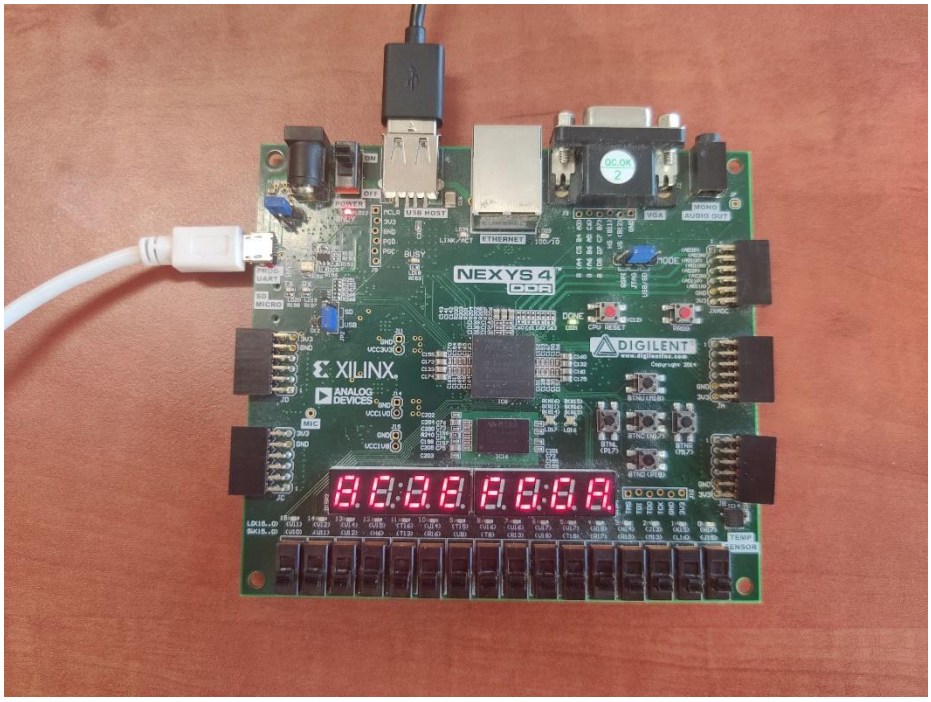

Figure 6

After pressing the backspace key the last character present on the segments will be deleted and the dot will be shifted with a position to the left(Figure 7).

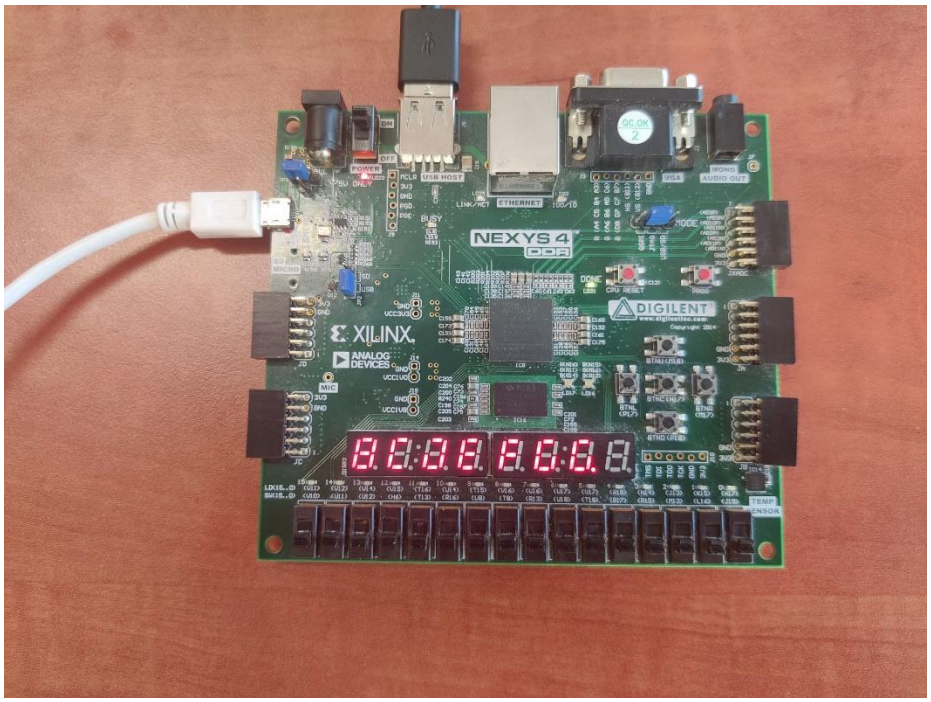

Figure 7

Another backspace pressed will have the same behaviour as we can see in Figure 8.

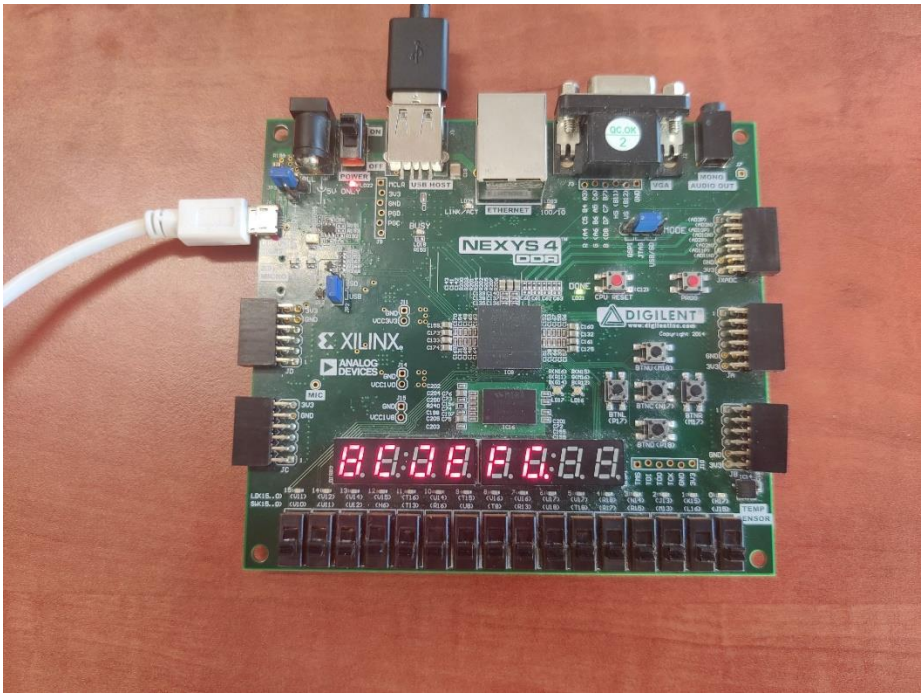

Figure 8

Further development possibilities

One more thing that could be done is to increase the size of the memory, such that, multiple lines of characters can be written just like in a real text editor. The up and down arrows would change the lines we see on the 7-segments displays. We could have also used the special key “Shift” in order to be able to write small letters as well.

Justification for using this solution

In terms of memory elements we decided to use a stack for our implementation, because it suits our needs. The way a text editor writes characters is similar to how a stack pushes an item. The backspace is actually just a pop. The stack has actually 16 addresses because the keyword we have used kept writing two characters at a time.