Lab 5: Creating Display Effects using 7-Segment Display

Shrey Patel 2019CS10400
Aayush Goyal 2019CS10452

May 2022

Contents

1 Implementation

1.1 Handling user input through push buttons

- The main goal of this part is to implement a functionality so that the user can control the brightness level of each individual digit. Now, both the 16-bit input and the 8-bit brightness(2-bit for each digit, so four levels of brightness per digit) are to be entered to the same 16 slide switches. So, whether the current configuration of slide switches is for the display digits or for setting brightness can be decided by suitable buttons.
- For this, we introduce a new signal 'next_d' which stores the next input to be displayed. Similarly, the next value of brightness is stored in the signal 'brightness'. The current input through the slide switches is pushed to 'next_d' on pressing the button 1(BTNL, W19) while its last 8-bits are pushed to the 'brightness' is pushed on pressing the button 2(BTNC, U18).

1.2 Debouncing

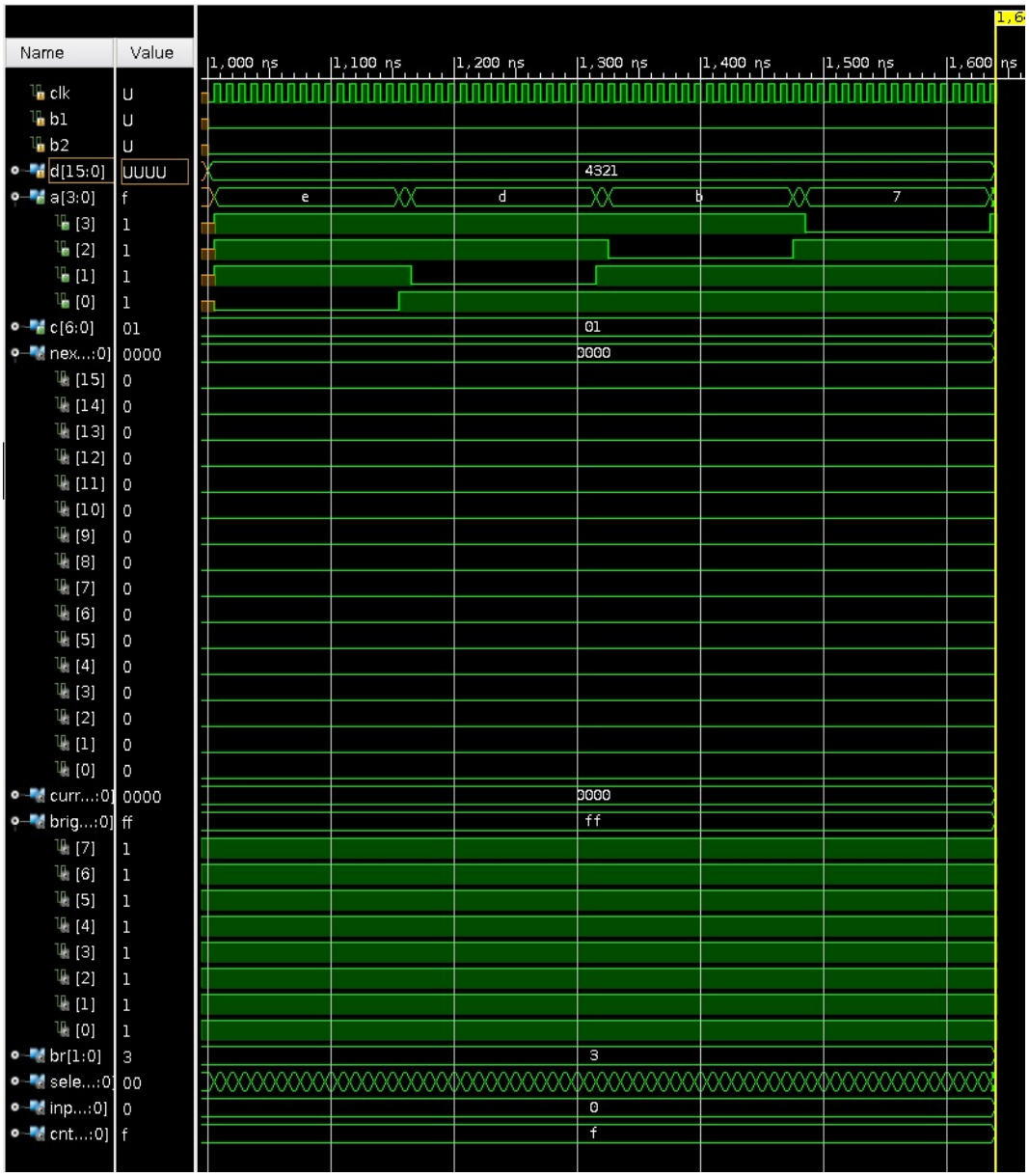
- Since we are using push buttons, there is naturally an issue of fluctuations in input due to irregularities in pushing the button by the user. So, a single button push is interpreted by the device as a series of fluctuations starting from 0 and finally ending at 1 till the time the user has kept the button pressed. And since the user typically presses the button for some milliseconds which is thousands of clock cycles, the high number of fluctuations in the device ports and registers may damage the device. This can be solved using debouncing.
- We discussed that the main problem is that the fluctuations are massive as seen by our original clock. So, we slow down the clock, so that the probability of those perturbations coinciding with the rising edge of this slow cycle is very low. So, even if the button input changes from 0 to 1 and back to 0 multiple times before fixing at 0, the device interprets this as a single transition from 0 to 1, and therefore only one assignment to a register/port. In this assignment, we have used a slow cycle which updates every 2^{20} cycles of the original clock. So, the time period of the slower clock is roughly 21 ms which is much larger than the duration of those perturbations, but still smaller than the amount of time for which the user keeps the button pressed. Thus, any button pressed is satisfactorily detected without any damage to the device.

1.3 Setting and Rotating Brightness

- The logic for implementing brightness using Pulse Width Modulation and rotation through 4 states is same as that in Assignment 4. But unlike in Assignment 4, we now need to fix the brightness with the digit itself and not with the digit position on the seven segment display.
- We did this by introducing a signal 'br' which stores the multiplexed value of the brightness signal depending on the current digit being displayed on the board. Now, the signal 'br' stores the brightness level out of 4 different values (00 being the darkest and 11 being the brightness).

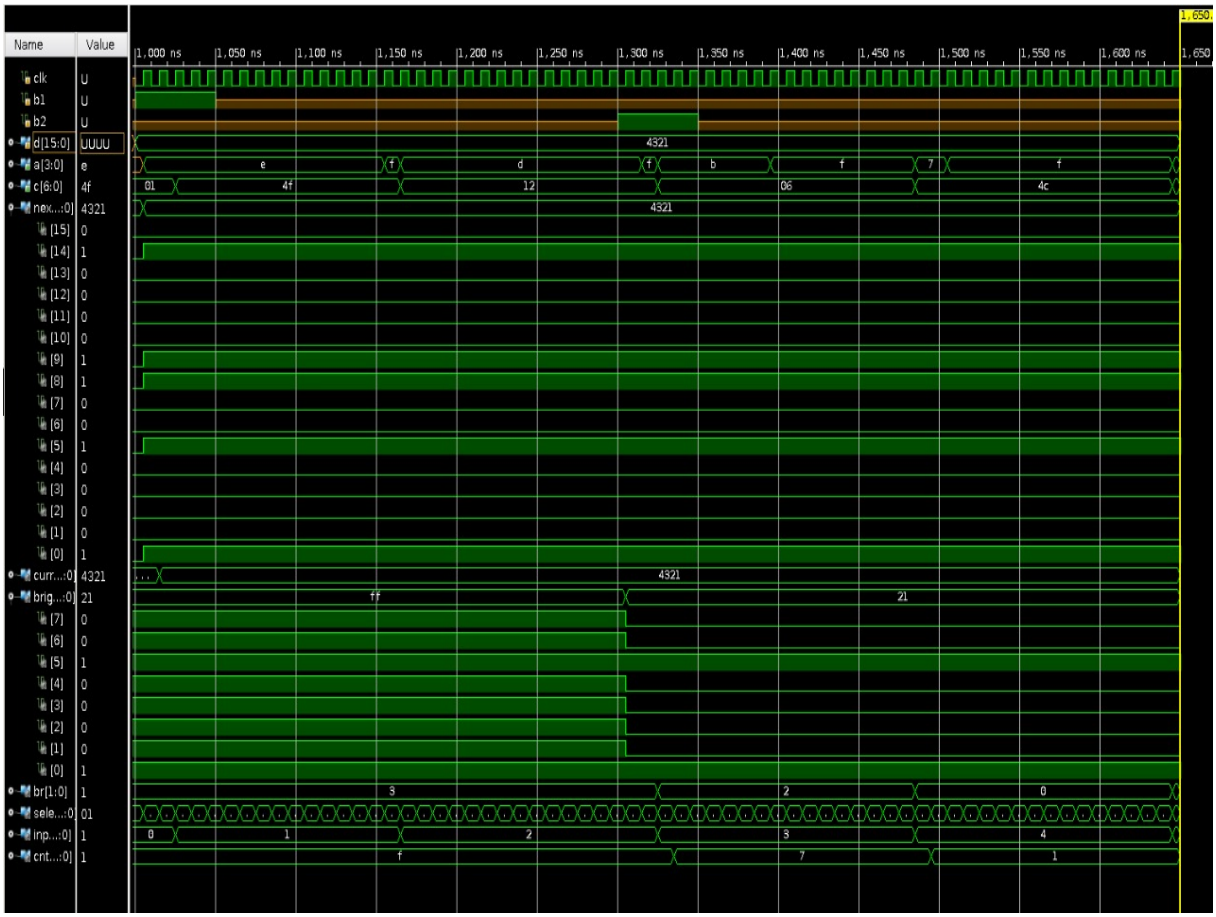
2 Simulation

2.1 Initial



- As shown in the above figure, even when the input 'd' is changed to 4321, the actual d values i.e. next_d and current_d remain zero since no button has been pressed. So, the seven segment display will show only 0s.
- Same occurs with the brightness which remains the highest i.e. 'ff' since no button has been pressed.

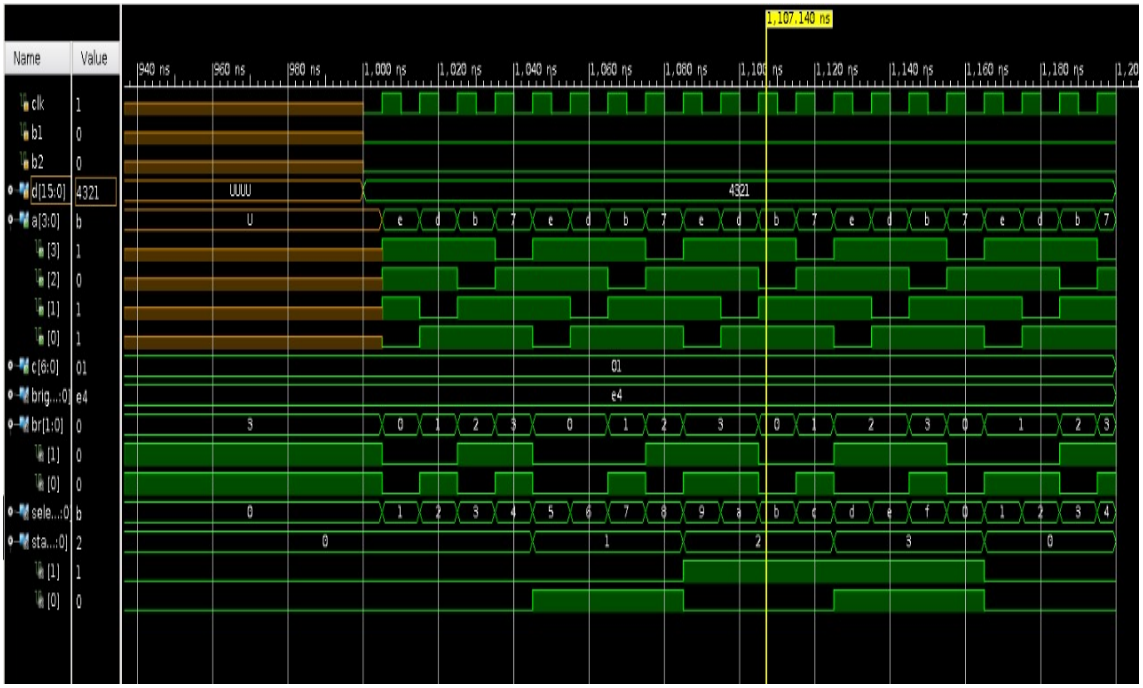
2.2 Button Presses



- In the above simulation, we can see that the button 1 is applied at time 1000 ns due to which the input 'd' which is 4321 is transferred to the signal 'next_d' which means that this updated value is now shown on the seven segment display of the board.
- Similarly, the button 2 is pressed at around 1300 ns, and due to this the brightness value will be updated to 21 which is precisely the last 8 bits of the input 'd' (4321). Thus, now this brightness change will be reflected on the board.
- So, this simulation shows that the two buttons are working exactly how they are intended to.

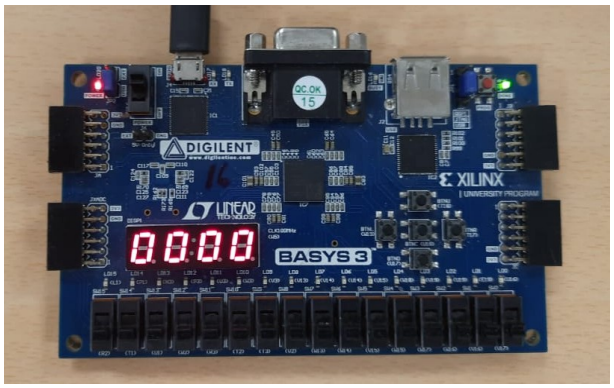
2.3 Rotating Brightness

- One significant difference from Assignment 4 is that now the brightness is associated with the digit itself instead of the position in the seven segment display.
- To show this, we first set the brightness value to "11100100" so that each digit now has a different brightness i.e. last digit has "00", 3rd digit: "01", 2nd digit: "10" and first digit: "11".
- From the simulation, we can see this initial configuration starting from 1000ns to 1040ns. From 1050ns, the state changes to "01". Earlier, the last digit '1' was mapped to the last digit of the seven segment display, now it is mapped to the 3rd digit after rotation. But we can in the interval 1060ns to 1070ns that this particular place i.e. anode position 1101 now has the brightness 00 while the anode position 1110 earlier had 00 brightness.
- From this, we can clearly see, that the brightness is always associated with the digit '1' regardless of the rotation state. The same can be seen for all the other three digits.

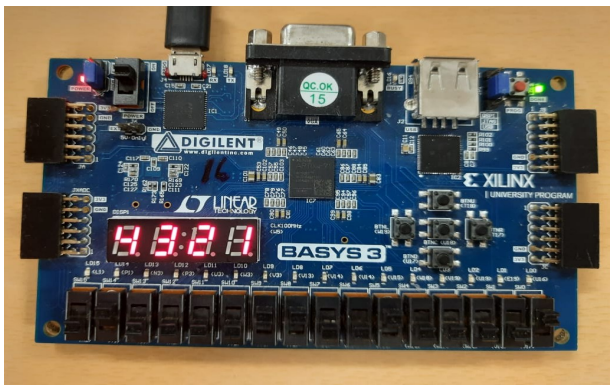


3 Output

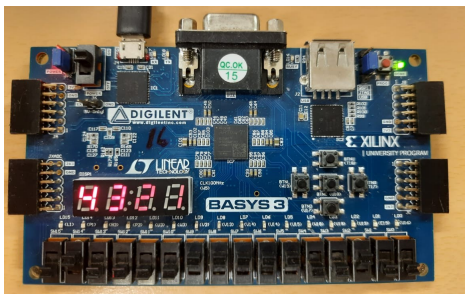
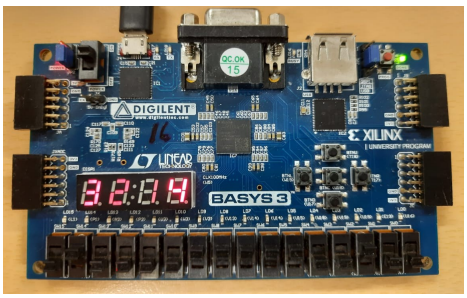
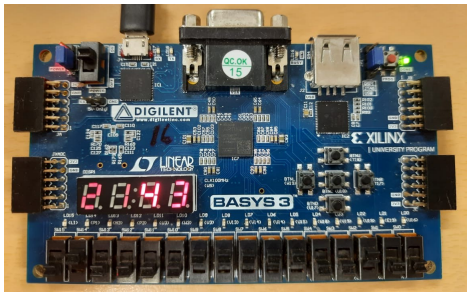
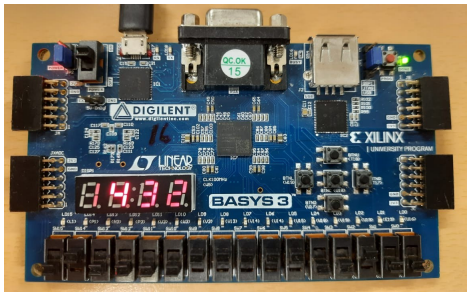
- Initially, display consists of all zeroes all of which are at the highest brightness of "11".



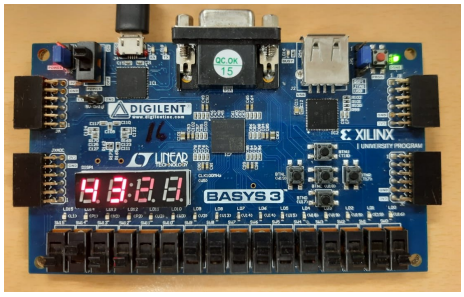
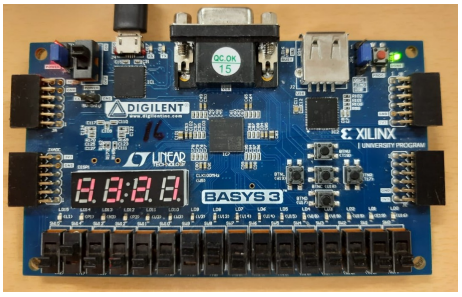
- Setting the slide switches to "0100001100100001" and pushing the button 1 causes the display to now show 4321, with no change in brightness yet.



- Setting the last 8 bits of slide switches to "11100100" and pressing the button 2 causes change in brightness of digits with 4 being brightest followed by 3, 2 and 1. The different rotations states in this case are as shown below:

- Some other examples of varying brightness

4 Resource Utilization

- **LUTs:** 31
- **Flip Flops:** 85
- **Block RAMs:** 0
- **DSPs:** 0