CS220 Lab#4
LCD Driver

Mainak Chaudhuri
Indian Institute of Technology Kanpur

Sketch

- Assignment#1: printing the string "WELCOME TO CSE, IIT KANPUR" on LCD
  - We will write the Verilog modules in such a way that they can be used in future for displaying other things as well
    - This is called a driver which is a generic interface between a hardware device (LCD in this case) and the software that uses the device
  - This assignment carries five marks

Lab#4

- Make a new folder Lab4_1 under CS220Labs to do the assignment
- Refer to lab#1 slides for Xilinx ISE instructions
- Finish pending assignments from lab#3 first

Assignment#1

- LCD (liquid crystal display)
  - Chapter 5 of the user guide explains the details of the LCD https://www.cse.iitk.ac.in/users/mainakc/2024Spring/lec220/ug230.pdf
  - First goal of this assignment is to understand the relevant portions of this chapter
  - The LCD screen can display two lines of characters and each line can have 16 characters
  - Figure 5-1 shows the interface between the LCD and the FPGA
    - FPGA can communicate four data bits (DB4, DB5, DB6, DB7) and three control bits (E, RS, R/W)

Assignment#1

- Interface between LCD and FPGA
  - Table 5-1 explains the meaning of the signals
    - Ignore all references to StrataFLASH
  - Take note of Figure 5-2
    - This will be needed in PlanAhead later
  - Before a character is displayed at a specific location of the screen, its ASCII code must be stored at an address of the display data RAM (DD RAM)
    - DD RAM is a small memory embedded inside the LCD
    - Figure 5-3 shows the mapping between LCD position and the DD RAM address
      - For example, to display a character at the 11th position from left in the second line, its ASCII code must be stored at DD RAM address 4A

Assignment#1

- Predefined characters
  - Most of the commonly used characters are already stored in a 16 x 13 character generator ROM (CG ROM)
    - The CG ROM contents are shown in Figure 5-4
    - The CG ROM address of a character is DB[7:0]
      - For example, the CG ROM address for the character 'S' is 0x53 where 5 represents the column number and 3 represents the row number
    - Interestingly, CG ROM address of a character is same as the ASCII/EBCDIC code of the character
      - When displaying a character, it is enough to look up Figure 5-4, find the CG ROM address of the character, and write that address in the correct location of DD RAM

Assignment#1

- Skip over the discussion on CG RAM
  - This memory is meant for storing user defined custom characters
    - We will not use this feature
- Set of commands for interacting with LCD
  - Table 5-3 lists the available commands for controlling the LCD
  - For any command to work, the LCD must be in enabled mode (i.e., LCD_E must be 1)
  - Each command sends a specific bit pattern to the LCD in the form of {LCD_RS, LCD_W, DB[7:0]}
    - Since only DB[7:4] are connected to the FPGA, DB[7:0] is broken down into two parts and sent one after another (DB[7:4] first and DB[3:0] next)

Assignment#1

- LCD commands
  - Pay attention to the following commands (we will be using these)
    - Clear display
    - Entry mode set: pay attention to I/D only
    - Display on/off
    - Function set
    - Set DD RAM address
    - Write data to CG RAM or DD RAM

Assignment#1

- Sending a command or data to the LCD
  - Figure 5-6 shows the general protocol for sending a command or data
  - It is important to observe the timing constraints
  - In general, LCD_RS, LCD_W, and the four data bits must be stable sufficiently earlier (at least 40 ns earlier) than setting LCD_E to 1
    - We will use a time gap of 20 ms
  - LCD_E should be kept 1 for at least 230 ns
    - We will keep it high for 20 ms
  - Before sending the next set of values for LCD_RS, LCD_W, and the four data bits, LCD_E must be 0 for at least 10 ns
    - We will wait for 20 ms after setting LCD_E to 0

Assignment#1

- Sending a command or data to the LCD
  - We will use the following general procedure
    1. Set LCD_E to 0
    2. Wait for 20 ms (1000000 cycles of 50 MHz clock)
    3. Send LCD_RS, LCD_W, and four data bits
    4. Wait for 20 ms
    5. Set LCD_E to 1
    6. Wait for 20 ms
    - Repeat steps 1 to 6 until all commands or data have been sent to LCD
  - If a communication requires sending eight bits of data, the most significant four bits are sent first (along with LCD_RS and LCD_W) and then the least significant four bits are sent (with the same LCD_RS and LCD_W)

Assignment#1

- Initializing the LCD
  - This is the first step before anything can be displayed
  - The initialization of LCD has a fixed sequence of commands; we will use the following sequence
    1. Set LCD_E=0 and wait for 20 ms
    2. Set LCD_RS=0, LCD_W=0, data=0x3, wait for 20 ms
    3. Set LCD_E=1 and wait for 20 ms
    4. Repeat steps 1 to 3 two more times
    5. Set LCD_E=0 and wait for 20 ms
    6. Set LCD_RS=0, LCD_W=0, data=0x2, wait for 20 ms
    7. Set LCD_E=1 and wait for 20 ms

Assignment#1

- Configuring the LCD
  - This is the second step before anything can be displayed
  - This step requires sending four commands to the LCD
  - The first one is the Function Set command which requires sending 0x28 as the data bits to the LCD
    - Set LCD_E=0 and wait for 20 ms
    - Set LCD_RS=0, LCD_W=0, data=0x2, wait for 20 ms
    - Set LCD_E=1 and wait for 20 ms
    - Set LCD_E=0 and wait for 20 ms
    - Set LCD_RS=0, LCD_W=0, data=0x8, wait for 20 ms
    - Set LCD_E=1 and wait for 20 ms

Assignment#1

- Configuring the LCD
  - The second command is the Entry Mode Set command which requires setting the data bits to 0x06 (see Table 5-3) so that DD RAM address is auto-incremented and shifting is disabled
    - Set LCD_E=0 and wait for 20 ms
    - Set LCD_RS=0, LCD_W=0, data=0x0, wait for 20 ms
    - Set LCD_E=1 and wait for 20 ms
    - Set LCD_E=0 and wait for 20 ms
    - Set LCD_RS=0, LCD_W=0, data=0x6, wait for 20 ms
    - Set LCD_E=1 and wait for 20 ms

Assignment#1

- Configuring the LCD
  - The third command is Display On/Off which requires setting the data bits to 0x0C (see Table 5-3) so that the display is on and the cursor and blinking are disabled
    - Set LCD_E=0 and wait for 20 ms
    - Set LCD_RS=0, LCD_W=0, data=0x0, wait for 20 ms
    - Set LCD_E=1 and wait for 20 ms
    - Set LCD_E=0 and wait for 20 ms
    - Set LCD_RS=0, LCD_W=0, data=0xC, wait for 20 ms
    - Set LCD_E=1 and wait for 20 ms

Assignment#1

- Configuring the LCD
  - The fourth command is Clear Display to prepare the display for your characters; requires setting the data bits to 0x01
    - Set LCD_E=0 and wait for 20 ms
    - Set LCD_RS=0, LCD_W=0, data=0x0, wait for 20 ms
    - Set LCD_E=1 and wait for 20 ms
    - Set LCD_E=0 and wait for 20 ms
    - Set LCD_RS=0, LCD_W=0, data=0x1, wait for 20 ms
    - Set LCD_E=1 and wait for 20 ms
- Now the LCD is ready to accept the characters you want to display

Assignment#1

- Displaying characters in the LCD
  - Since we have enabled auto-increment of DD RAM address, if we send the starting address of the DD RAM, each character will automatically increment the address
  - The command to set the starting address is Set DD RAM address
  - We will start displaying from the leftmost corner of the first line
    - This corresponds to DD RAM address 0x00
    - The corresponding Set DD RAM address command requires setting the data bits to 0x80 (see Table 5-3)

Assignment#1

- Displaying characters in the LCD
  - Set DD RAM address to 0x00 using command 0x80
    - Set LCD_E=0 and wait for 20 ms
    - Set LCD_RS=0, LCD_W=0, data=0x8, wait for 20 ms
    - Set LCD_E=1 and wait for 20 ms
    - Set LCD_E=0 and wait for 20 ms
    - Set LCD_RS=0, LCD_W=0, data=0x0, wait for 20 ms
    - Set LCD_E=1 and wait for 20 ms
  - Now we are ready to send the characters for the first line of the display

Assignment#1

- Displaying characters in the LCD
  - We will assume that the first line is stored in a 128-bit array where each character is a byte corresponding to its ASCII value or equivalently its CG ROM address (see Figure 5-4)
    - Each character corresponds to DB[7:0]
    - Use command Write Data to CG RAM or DD RAM (see Table 5-3)

    1. Set LCD_E=0 and wait for 20 ms
    2. Set LCD_RS=1, LCD_W=0, data=DB[7:4], wait for 20 ms
    3. Set LCD_E=1 and wait for 20 ms
    4. Repeat steps 1 to 3 with data=DB[3:0]

    - Repeat steps 1 to 4 for each character in the first line

Assignment#1

- Displaying characters in the LCD
  - Once the characters of the first line are sent, we need to set up the LCD for a line break
  - This is done by setting the DD RAM address to the beginning of the second line i.e., address 0x40 (see Figure 5-3)
  - This requires sending a Set DD RAM address command with data bits 0xC0 (see Table 5-3)
    1. Set LCD_E=0 and wait for 20 ms
    2. Set LCD_RS=0, LCD_W=0, data=0xC, wait for 20 ms
    3. Set LCD_E=1 and wait for 20 ms
    4. Repeat steps 1 to 3 with data=0x0

Assignment#1

- Displaying characters in the LCD
  - Use the same method as used for the first line for sending the second line characters to the LCD
  - At the end make sure to set LCD_E=0
- Planning the Verilog modules
  - We will use two modules
  - The first module (LCD_driver) takes as input the two lines of characters (each a 128-bit array), and clk
  - The first module's outputs are LCD_RS, LCD_W, LCD_E, and the four data bits
  - The module LCD_driver implements the entire algorithm for displaying the characters of the two input lines

Assignment#1

- Planning the Verilog modules
  - Structure the module LCD_driver is as follows
    - Have a counter that can count up to 1000000
    - On posedge clk, if the counter has reached 1000000, execute one step of the LCD algorithm; else increment the counter
    - To determine which step of the LCD algorithm needs to be executed now, you need to maintain a state register that gets incremented after each step of the algorithm
    - Your algorithm will have four distinct parts
      - Initialization and configuration of the LCD
      - Displaying first line
      - Line break
      - Displaying second line

Assignment#1

- Planning the Verilog modules
  - The top-level module simply instantiates LCD_driver
    - The top-level module's input is clk and outputs are LCD_RS, LCD_W, LCD_E, and the four data bits
      - These will be connected to the appropriate pins
    - The instantiation of LCD_driver passes two strings, clk, LCD_RS, LCD_E, and the four data bits
      - Make sure that each string has exactly 16 characters (pad at the end with whitespaces to make 16 characters)
      - For example: "WELCOME TO CSE, ", "IIT KANPUR   "
- Use PlanAhead to assign pins to clk, LCD_RS, LCD_W, LCD_E, and the four data bits
  - You should be using the clock generated at pin C9 for clk (Figure 3-2 of Chapter 3)

Assignment#1

- Synthesize the hardware and verify that the LCD driver is working correctly
- Try changing the strings in the top-level module and see if the driver works