Lehrstuhl für Integrierte Systeme **Technische Universität München**

Project Lab IC-Design

Projektpraktikum IC-Entwurf

Summer Semester 2017

Iffat Brekhna

Matriculation Number: 03670102

Contents

1 Display Module

1.1 Introduction

The display module receives input lines from all other modules in BCD format. It has a display driver which is responsible for initializing the LCD using the control signals. The display module decides which data should be displayed at a certain time, in other words, it forward the data of the mode the clock module is in. The current mode input is received from the global Finite State Machine (FSM) as a 3 bit input. Figure 1 shows the functional specification of the display module. There was no issue with managing the timing issues in this module because our clock module is running with a system clock (clk) of 10KHz. This makes the time period to be 1000us which is much larger then the minimum timing constraints given in the data sheet of the LCD for the operations that we do.

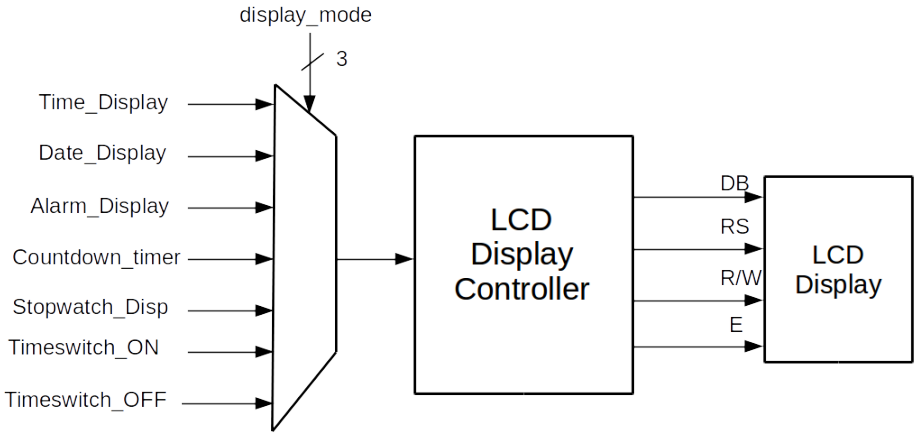

Figure 1: Functional specification

1.2 Implementation

The register transfer level (RTL) logic of the display driver is shown in Figure 2. The internal signals are defined below:

- *line* : selects the lines to be saved in the register file
- *r_en* : enable for register in write cycle
- *lines_done* : signal to notify the LCD FSM when all the words of the 2 lines from the register file are written into the DDRAM in the LCD module
- *tx_done* : signal from the control FSM to notify whenever a single word is written into the DDRAM of the LCD
- *addr_mux* : address of the word to read from the register file
- *data_out* : output from the register file i.e data to be displayed in the LCD

- fsm_data : control data from the LCD FSM
- $Data$: The data to be written into the LCD DDRAM
- RS : Register select to determine whether it is a instruction entry or a data entry
- R/W : It is always tied to 0 because we are always only writing data to the LCD DDRAM
- E : Enable for sampling of data

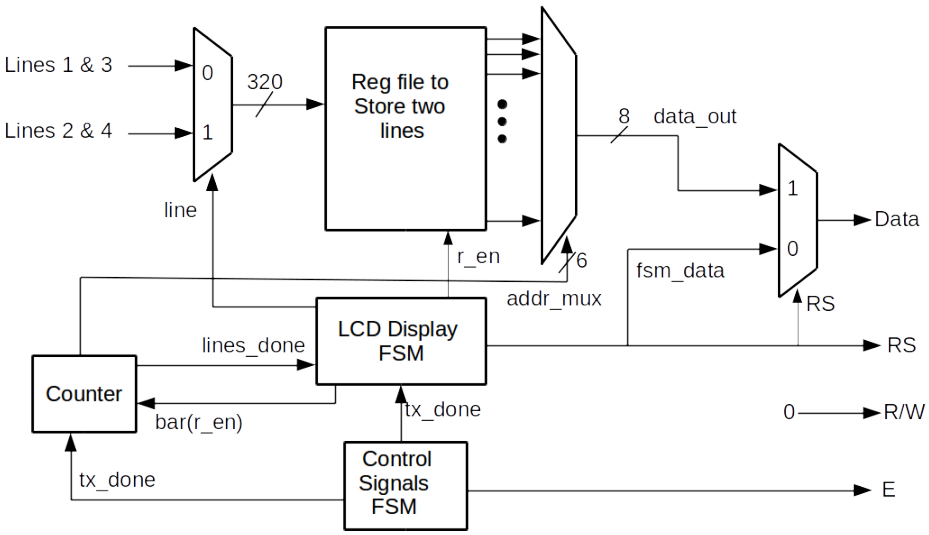

Figure 2: Register Transfer Logic of the display driver

1.2.1 Register File

The Register File's size is 320 bits so that it can save two lines at a time. The input to this register file is either line 1 concatenated with line 3 or line 2 concatenated with line 4 because in the LCD module the DDRAM address space is divided in a way that line 1 address is followed by line 3 address and line 2 address is followed by line 4 address. So it's easier to write data in the DDRAM in this way rather than writing one line, changing the DDRAM address and then writing another line for all four lines.

1.2.2 LCD Display FSM

Figure 3 shows the FSM diagram of this block. It is responsible for the initialization phase of the LCD and also the display functionality. The signal 'D' defined in this module is the same signal defined as fsm_data in the RTL logic of the display driver. It was shortened to D due to space issues while drawing the FSM in LaTeX. Each state is defined below.

- S_0 : It is the first step of LCD initialization in which the function set step is executed. D = 0x38 means setting a 8-bit bus mode with MPU, having a 2-line display mode and a 5x8 dot format display mode set.

- S_1 : It is the second step of LCD initialization in which the display is turned on. D = 0x0C means that the entire display is turned on, the cursor is off and blink is off.
- S_2 : It sets the DDRAM address. D=0x80 means the address is set to line 1.
- S_3 : It writes the data from the register file into the LCD DDRAM. D=X means that it can be any data depending on the input from the outside (other modules). The FSM continues to be in this state unless both the lines (line 1 and 3) are written into the DDRAM i.e it switches to the next state when $lines\_done = 1$.
- S_4 : It sets the DDRAM address. D=0xC0 means the address is set to line 2.
- S_5 : It writes the data from the register file into the LCD DDRAM. D=X means that it can be any data depending on the input from the outside (other modules). The FSM continues to be in this state unless both the lines (line 2 and 4) are written into the DDRAM i.e it switches to the next state when $lines\_done = 1$.

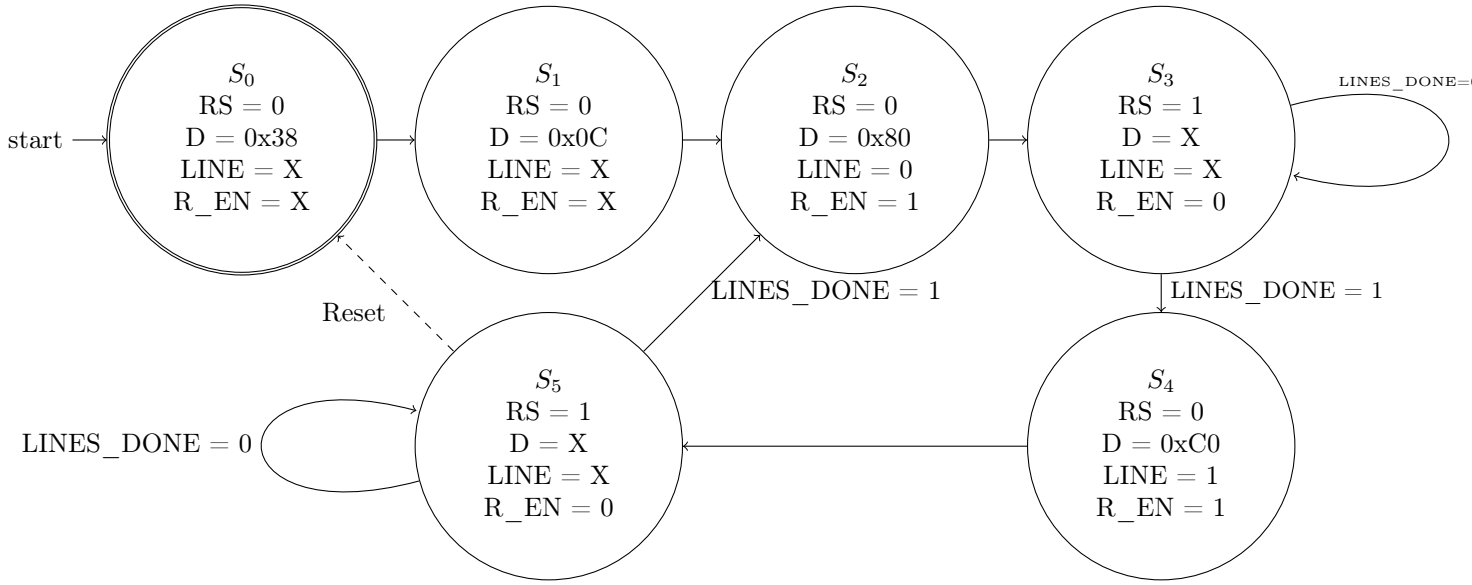

Figure 3: LCD Display FSM. All transitions not labeled occur on CLK↑ if $tx\_done = 1$

1.2.3 Control Signals FSM

Figure 4 shows the FSM diagram of this block. It is responsible for generating the control signals which are inputs to the LCD FSM, the counter and it also generates the 'E' signal of the LCD. The tx_done signal represents the transmission of a single block of data (one character) to the LCD DDRAM.

1.2.4 Counter

The counter is incremented when $r\_en = 0$, $tx\_done = 1$ and $RS = 1$. When the count value equals 39 the signal $lines\_done = 1$ implying that all 40 characters have been written into the DDRAM of the LCD. Also, the $addr\_mux$ signal is calculated by subtracting the count value of the counter from 39 to take care of the order of reading data from the register file.

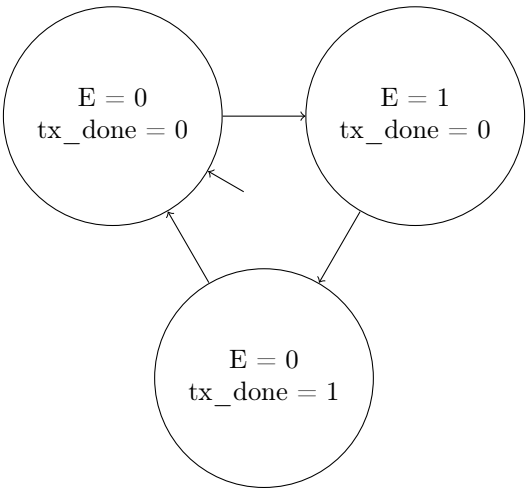

Figure 4: Control Signals FSM. All transitions occur on CLK↑

2 Changes from the Initial Presentation

In the initial presentation, I presented some things which while writing the code I realized had to be changed. First of all, the functional specification of the display module as shown in figure 1 has been changed. In the initial presentation it was proposed that data is received from other modules as *std_logic_vectors*. However, this format has been changed to BCD. Also initially, the time display was always forwarded to the display driver regardless of the state the clock module was in but later this logic was also changed as the Time Switch states had to be taken care of.

Secondly, the register transfer level (RTL) logic of the display driver as shown in figure 2 has been modified. In the previous version there was no mux in front of the register file to read each word. An issue arose while reading from the register file and the order to read in as the display was displaying everything in the wrong order. To solve that issue, a mux was added and the select line of the mux (*addr_mux*) was set in such a way that it reads the line from left to right i.e in the order the line has to be displayed.

Thirdly, the *LCD_DISPLAY_FSM* module as shown in figure 3 also had a logical error. In the initial presentation I proposed turning ON the LCD first and then setting the mode but after several tries of failing to run the LCD I found out that the order was wrong, I was supposed to set the mode first and then turn the LCD ON. So the order of the first two states changed.

3 Results

Figure 5 shows the simulation results of the LCD FSM module. Figure 6 shows the simulation results of the Control FSM module. Figure 7 shows the simulation results of the Display Driver. The LCD is running perfectly fine in real life as well (as demonstrated in the demo).

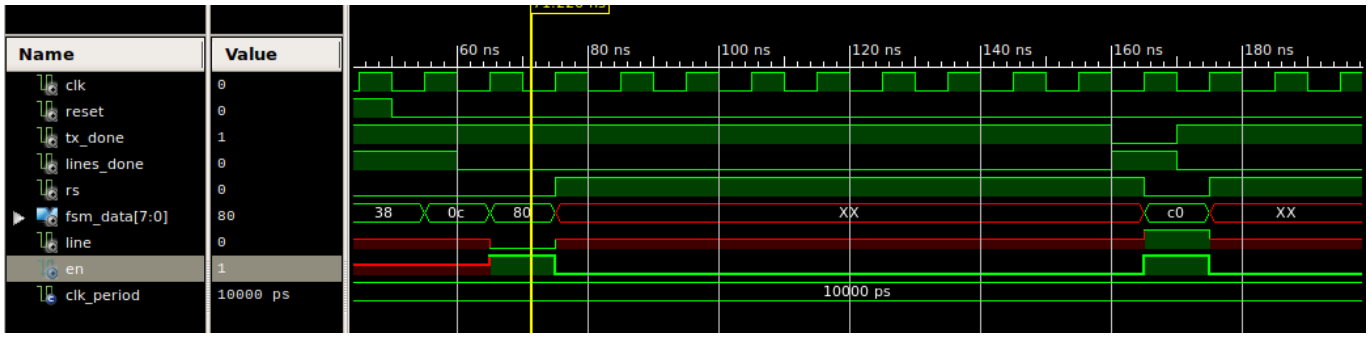

Figure 5: Simulation Results of the LCD FSM Module

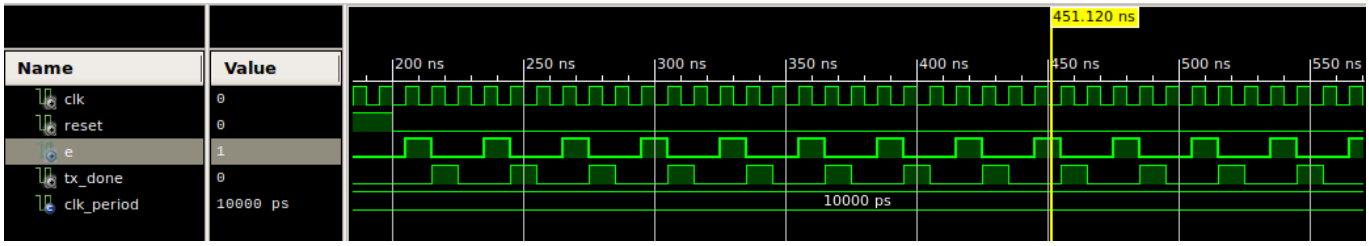

Figure 6: Simulation Results of the Control FSM Module

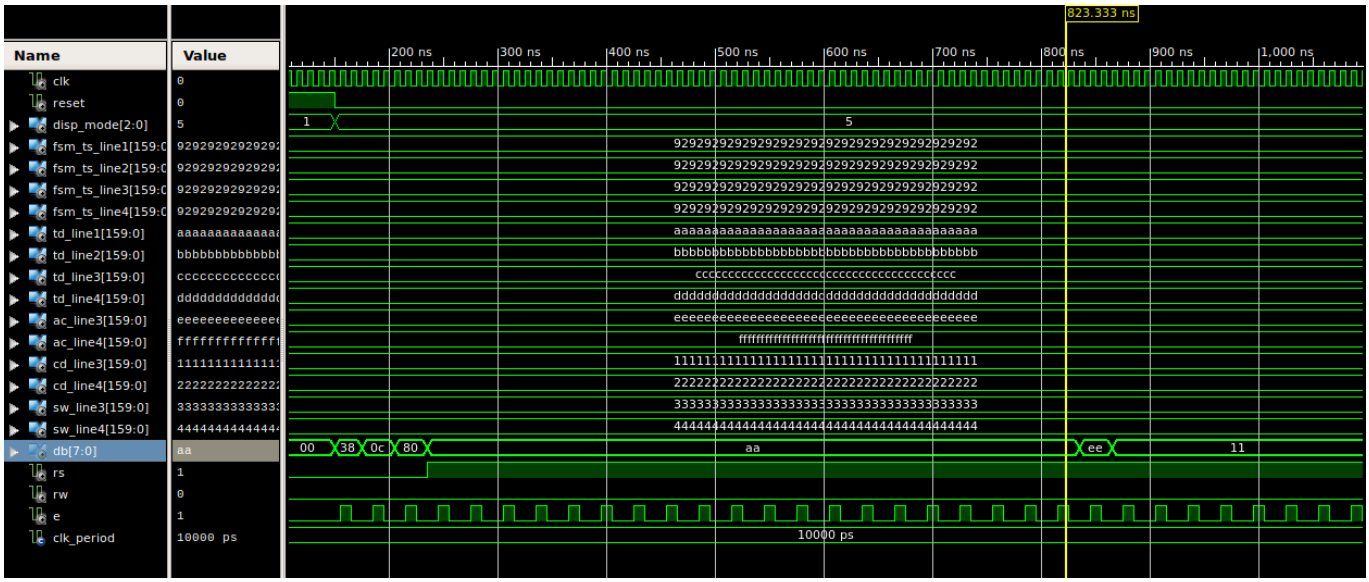

Figure 7: Simulation Results of the Display Driver