FPGA Interfacing of HD44780 Based LCD Using Delayed Finite State Machine (FSM)

Edwin NC Mui
Custom R & D Engineer
Texco Enterprise Ptd. Ltd.

{blackgrail2000@hotmail.com}

Abstract

This paper presents a novel method of interfacing the FPGA to a HD44780 based Text LCD based on using delay elements with a Finite State Machine (FSM). There exist many pre-written source codes and functions for microprocessor and microcontroller based implementations where the user will only have to call the functions perform a read or write operation on the HD44780. However, few to little sources actually place focus on interfacing the HD44780 to a programmable logic device, such as an FPGA, to the HD44780. Hence, in this paper, a simple FSM is devised to create a simple controller within the FPGA to output text onto the HD44780 based Text LCD. The process undergone to design the FSM and its additional control hardware is illustrated in a step-by-step manner. The FSM is then implemented on a XESS XSA-200 board which houses the Xilinx Spartan-2 XCS200 FPGA. The Place and Route report from the implementation indicates that the design occupies 87 slices with a maximum achievable clock frequency of 108.98 MHz.

1. Introduction

For various electronics based projects ranging from hobbyist electronics to industrial based projects, the HD44780 based Text LCD is one of the most preferred choice when it comes to selecting a display output unit.

Typically, the HD44780 is interfaced to a microcontroller (μC) or a microprocessor (μP) in order to generate text on the LCD. This is the most cost and time effective implementation since one can easily generate the signals needed in order to write to the LCD simply by writing a program in assembly or other higher level languages, such as C. Also, this task is made even easier with the availability of large amounts of pre-written codes and libraries, allowing hobbyists and engineers to control the HD44780 based Text LCD directly by just calling the function when writing the program.

On the other hand, to interface the HD44780 based LCD to a programmable logic device such as a CPLD or an FPGA would require a design of higher complexity in terms of hardware as opposed to using a microcontroller which is software based. For the case of using programmable logic device, to generate the signals and waveforms required by the HD44780 would require extremely deep amounts of logics. An easier way to go about this is to have a Finite State Machine (FSM), which acts as a controller to control the timing and generation of the necessary signals to communicate with the HD44780. The section that follows explains the process that was undergone to produce a FSM using delay elements to accomplish this simple design for interfacing an FPGA to a HD44780 based Text LCD.

2. HD44780 Operational Characteristics

This section illustrates the electrical, timing and control characteristics of the HD44780 Based Text LCD. Only the most relevant sections of the HD44780 characteristics for this paper are included here in a summarized manner. For the complete specifications and functionality of the HD44780 controller, please refer to its datasheet.

2.1. Electrical Characteristics of the HD44780

From [1], it is stated that for a given HD44780 V_{CC} input of 4.5V to 5.5V, the absolute minimum and maximum voltage levels for both logic ‘1’ and ‘0’ are 2.2V and 0.6V respectively. For logic ‘0’, it will not be a problem since the FPGA will just have to pull the voltage to the ground level. For logic ‘1’, most FPGAs will not have a problem fulfilling the requirement that the voltage level must be at least above 2.2V. This is made possible since most modern FPGAs are able to operate in Low Voltage Transistor-Transistor Logic (LVTTL) mode, where the voltage on the output pins of the FPGA are typically 3.3V if the FPGA is powered by 3.3V on the FPGA’s V_{CC} line, which is more than enough to satisfy the minimum voltage requirement.

2.2. Timing Characteristics of the HD44780

The timing characteristics for the HD44780 for a write operation can be acquired from [2] and it is reproduced in the Figure 2.2 below. It should be noted that the timing diagram here is not drawn to scale and it is just to show the important setup times in order to have the FPGA to drive the HD44780.

While a more detailed timing diagram is provided in [1], the author of [2] has simplified the timing diagram such that it is easier for to interpret. In the Figure 2.1, the author considers the case where both the RS bit and the 8-bit Data are simultaneously written, as compared to [1], where the RS bit and 8-bit Data are applied at different times. By writing both the Data and RS bit at the same time, one can account for the setup time for RS and Data, rather than accounting for them separately if both RS and Data were written at different times.

In this paper, only the write operation is used for the FPGA hardware to communicate with the HD44780. Due to this, the RW pin is always tied to the ground. Therefore, the timing characteristic for the RW pin can be ignored for further simplification of the timing diagram in Figure 2.1. With this in mind, one would only need to work out the necessary hardware for generating the 40ns, 230ns, and the execution time delay for the HD44780 instruction being issued. The 10ns delay can be safely ignored, since a large delay will have to be generated after the write operation in order to ensure that the process is successfully completed, before beginning with another write operation. Such large delays are usually at least 40us, which is exceedingly more than enough to account for the 10ns delay.

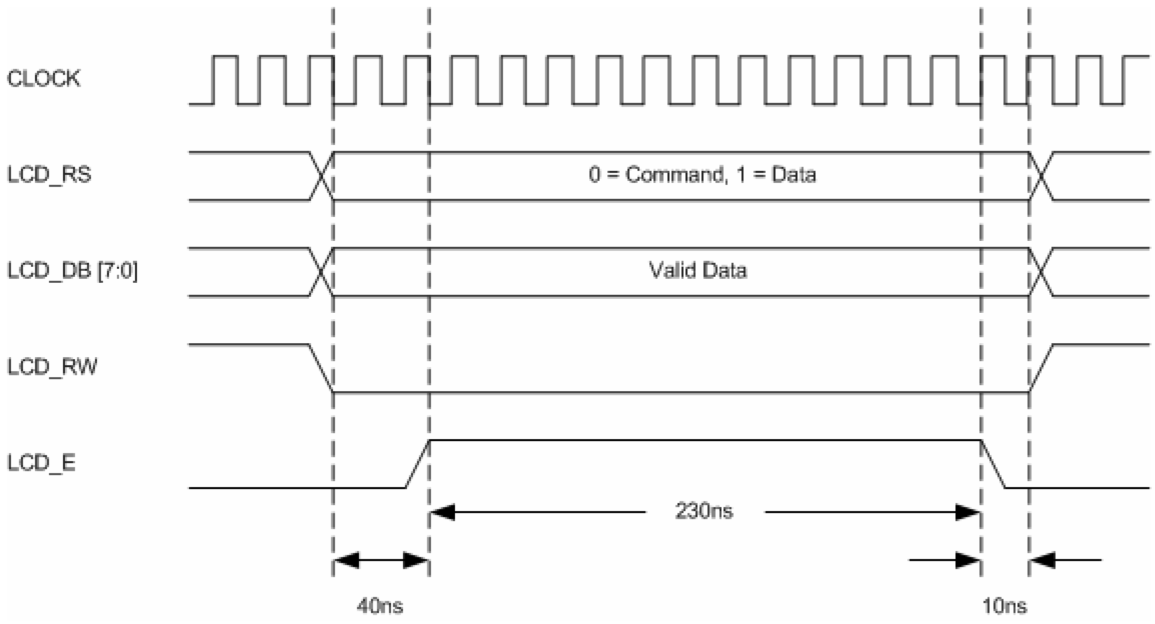

Figure 2.1. A simplified timing diagram for a write operation from [2].

2.3. Control Characteristics of the HD44780

Again, only the most relevant sections of the control characteristics from the HD44780 datasheet are presented here. For further details on the control of the HD44780, please refer to its datasheet.

In this paper, the following instructions of the HD44780 were used and they are tabled below in Table 2.1.

Function	**RS**	**R/W**	**Data Bits**							
			DB7	**DB6**	**DB5**	**DB4**	**DB3**	**DB2**	**DB1**	**DB0**
Clear Display	0	0	0	0	0	0	0	0	0	1
Display On/Off	0	0	0	0	0	0	1	D	C	B
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S
Function Set	0	0	0	0	1	DL	N	F	-	-
Write Data	1	0	D7	D6	D5	D4	D3	D2	D1	D0

Table 2.1. Instruction set used for the FPGA implementation.

The control bits are listed as follows in the table below.

Display On/Off	Entry Mode Set	Function Set
D = 1, Display on. D = 0, Display off. C = 1, Cursor is displayed. C = 0, Cursor not displayed. B = 1, Blinking enabled. B = 0, Blinking disabled.	I/D = 1, Cursor moves right. I/D = 0, Cursor moves left. S = 1, Display shift enabled. S = 0, Display shift disabled.	DL = 1, 8 bit data length. DL = 0, 4 bit data length. N = 1, 1/16 Duty N = 0, 1/8 Duty F = 1, 5x10 Dots F = 0, 5x7 Dots

Table 2.2. Control bits for selected HD44780 instructions.

Also from the HD44780 datasheet the execution time for the instructions in Table 2.3 above is also included below.

Function	Execution Time
Clear Display	1.52ms
Display On/Off	37us
Entry Mode Set	37us
Function Set	37us
Write Data	37us

Table 2.3. Execution time for selected HD44780 instructions.

2.3.1. HD44780 Initialization

The initialization sequence can be summarized in the order as shown below as taken from the HD44780 datasheet. This process is necessary should the power up conditions for the automatic initialization is not fulfilled. However, assuming the worst case scenario, it would be safer for perform the initialization sequence when interfacing the HD44780 to the FPGA, as illustrated in Figure 2.2 below. The Display OFF instruction can be replaced with a Display ON/OFF instruction that sets the bit to turn the display on, enable the cursor and its blinking.

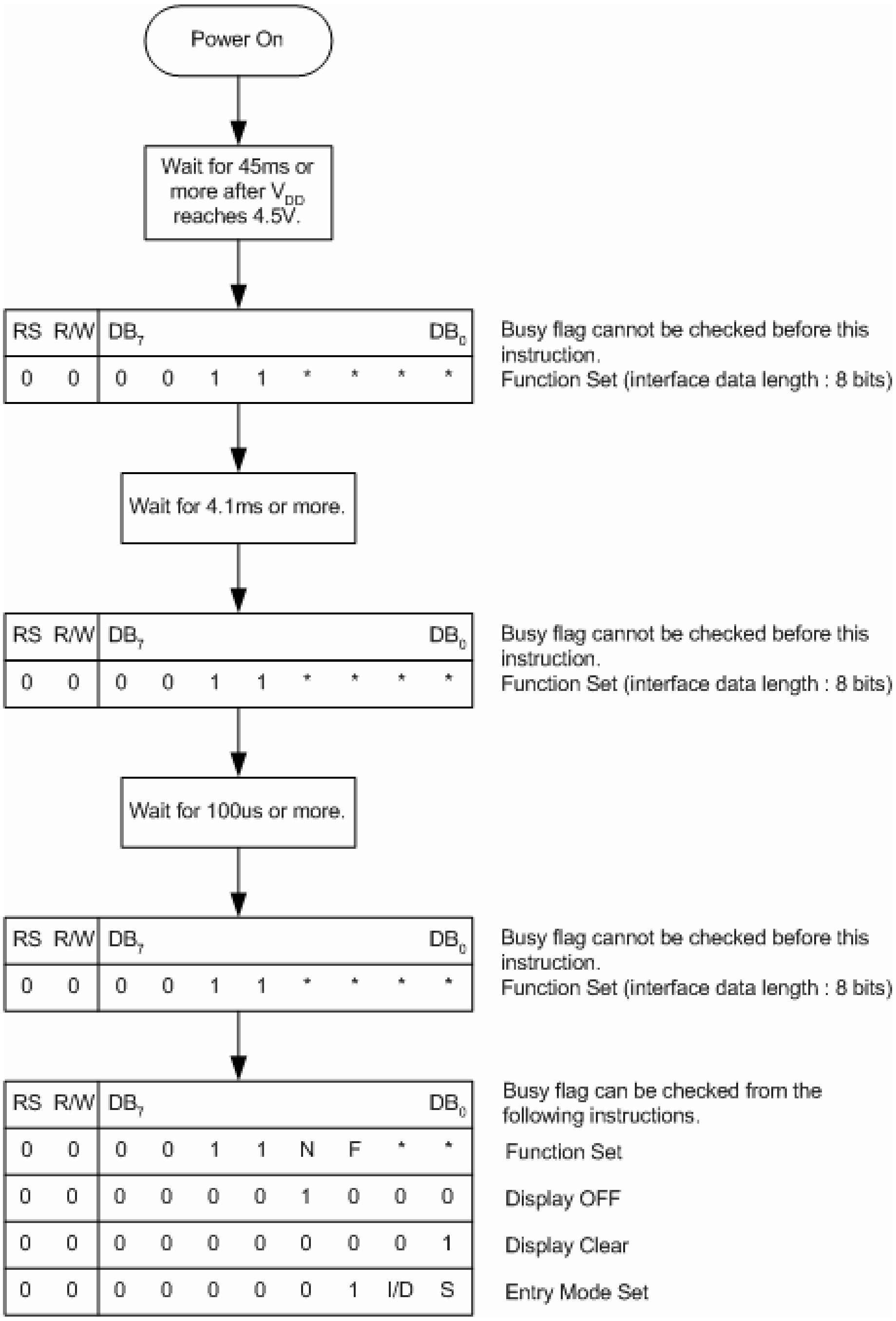

Figure 2.2. Initialization sequence for the HD44780 when power supply conditions are not satisfied. [1]

3. FSM and Hardware Layout

For the purpose of demonstrating the interface between the FPGA and the HD44780 based LCD, the following process flow will be used.

- Wait 45ms.
- Write 0x38, wait 4.1ms.
- Write 0x38, wait 100us.
- Write 0x38, wait 40us.
- Write 0x38, wait 40us.
- Write 0x0F, wait 40us. (Display ON, Cursor ON, Blinking ON)

- Write 0x01, wait 1.64ms. (Clear display)
- Write 0x06, wait 40us. (Entry Mode Set, Auto-Increment, No Shift)
- Write ASCII ‘H’, wait 40us.
- Write ASCII ‘e’, wait 40us.
- Write ASCII ‘l’, wait 40us.
- Write ASCII ‘l’, wait 40us.
- Write ASCII ‘o’, wait 40us.
- Write ASCII ‘ ’, wait 40us.
- Write ASCII ‘a’, wait 40us.
- Write ASCII ‘l’, wait 40us.
- Write ASCII ‘l’, wait 40us.

It can be observed from the above that there are exactly 16 write operations. In this case, it is possible to round up all of the 16 write instructions and have them tied to a 16 to 1 multiplexer. Also, a 4-bit counter can also be used to sequentially select the instructions according to the prescribed process flow above. Figure 3.1 below shows the hardware diagram for both the counter and the 16 to 1 multiplexer, which in essence forms the instruction counter. The counter is controlled by a Chip Enable (CE) signal in order to ensure that the *Inst_Cnt* selects the correct instruction at any given time. The CE signal is asserted high when the Finite State Machine (FSM) reaches the *Load_Next_Data* state.

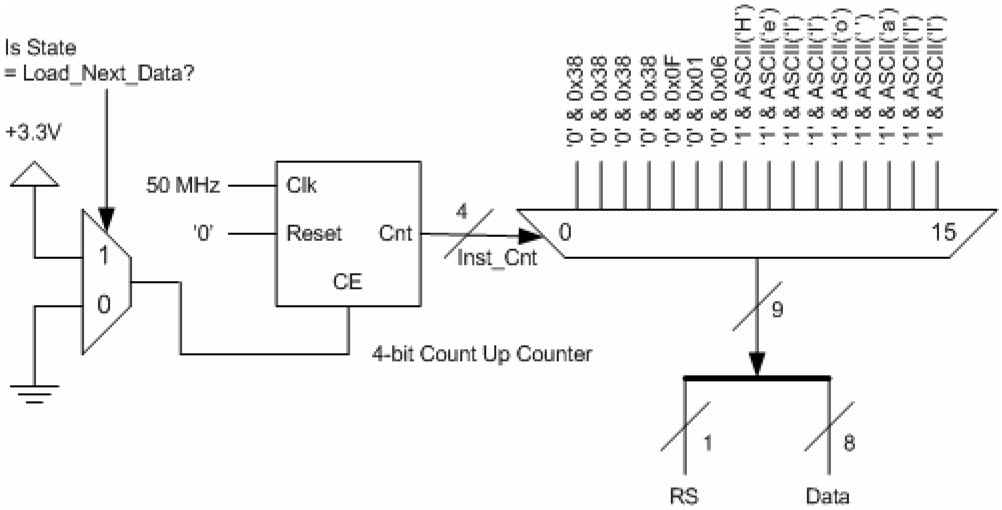

Figure 3.1. Instruction counter for relaying command/data to the HD44780 LCD.

Following this, a Finite State Machine (FSM) is devised to ensure the process flow above is followed. Figure 3.2 below shows the FSM devised in accordance to the above process flow.

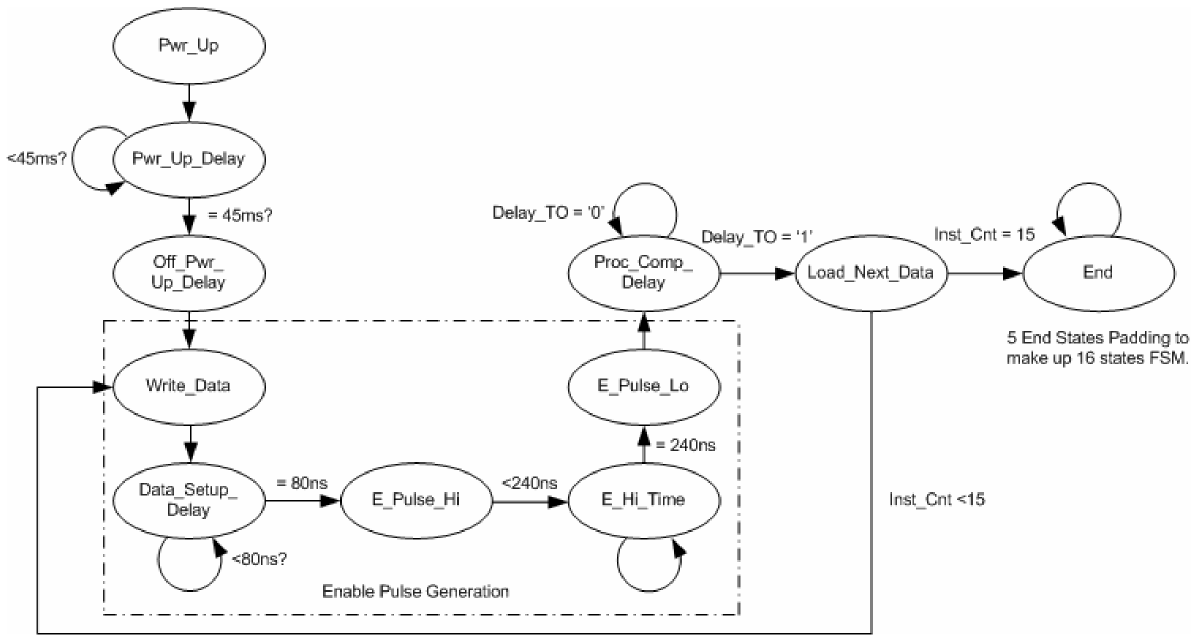

Figure 3.2. The Finite State Machine for interfacing the FPGA with the HD44780 based LCD.

Following the *Pwr_Up* state which happens as soon as the power is applied to the circuit, the FSM enters the *Pwr_Up_Delay* state in which it will wait for 45ms before proceeding to the next state. For a given clock input of 50 MHz, or period of 20 ns, 45ms would translate to 2,250,000 clock cycles. To implement this delay element, a 21-bit count-up counter with a Chip Enable (CE) signal is used. A 21-bit count-up counter is used since 2^{21} = 2,097,152, is approximately close to 2,250,000 clock cycles. The number of bits required by subsequent delay elements that will be mentioned will also be computed in the same manner.

The hardware diagram for this implementation is shown below in Figure 3.3. A multiplexer is use to switch CE between output '1' (3.3V) and '0' (Ground), depending on the state. The CE signal is asserted high during the *Pwr_Up* and *Pwr_Up_Delay* states, to have the counter counting up during these two states. On other states, CE will be driven low, and the clock pulses are ignored by the counter. When the delay element counter finally reaches its maximum value, the FSM will switch to the *Off_Pwr_Up_Delay* state. The propagation to the *Off_Pwr_Up_Delay* will automatically disable the counter.

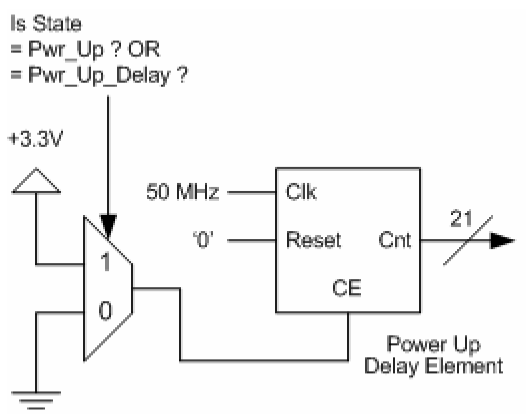

Figure 3.3. Hardware implementation of the Power Up Delay.

From the *Off_Pwr_Up_Delay* state, the FSM switches to the *Write_Data* state, entering the Enable pulse generation state machine. The Enable pulse generation state machine is formulated such that it would comply with the timing and the waveforms shown in Figure 2.1 above. From the *Write_Data* state, the FSM proceeds to the *Data_Setup_Delay* state, where a delay is generated to ensure that the setup time prior to the rising edge of the Enable pulse is sufficient. From [2], it is stated that the minimum setup time is 40ns. Therefore, technically, a 1-bit counter will be sufficient, given an input clock of 50 MHz. However, a 1-bit counter is unrealistic for practical implementations hence a 2-bit counter is used instead. As the result, 80ns of delay is generated instead. A counter with CE similar to that in Figure 3.3 above is used. Figure 3.4 below shows a circuit for controlling the CE of the 2-bit count up counter.

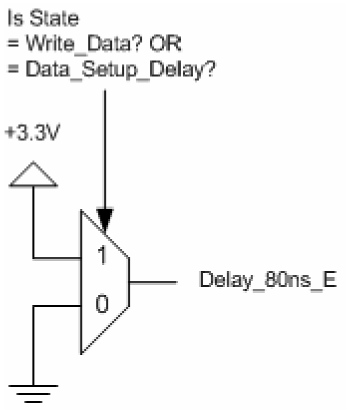

Figure 3.4. Hardware implementation of the circuit for controlling CE for the 80ns delay element.

An Enable pulse must be at least 240ns to be considered valid by the HD44780. This is taken care of by the next 3 states, namely, *E_Pulse_Hi*, *E_Hi_Time* and *E_Pulse_Lo*. A 4-bit counter is used as a delay element to ensure that the Enable pulse width is greater than 240ns. The figure below illustrates the hardware implementation for generating the Enable pulse. The E wire which is connected to the E pin of the HD44780 LCD is controlled by the FSM. The FSM sets the pin high during *E_Pulse_Hi* and *E_Hi_Time* states while keeping it low during other states. Also, the delay element is only enabled during the same states, which results in the Enable pulse being generated precisely for the duration of the delay element, as shown in the timing diagram in Figure 2.1 above.

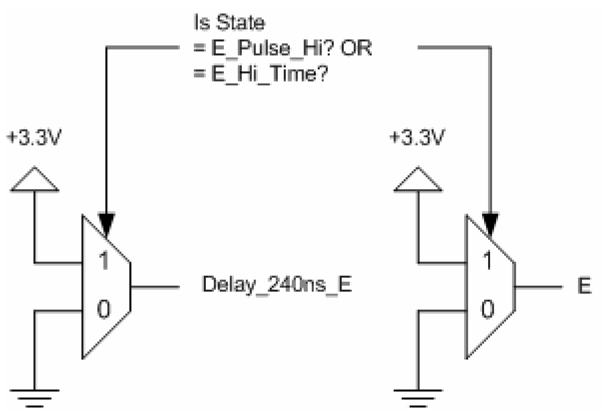

Figure 3.5. Circuits for generating an Enable pulse for a duration of at least 240ns.

From *E_Pulse_Hi* state, the next state that follows is *E_Pulse_Lo* which will automatically drive the E pin low. After this, the FSM enters the *Proc_Comp_Delay* state which activates the corresponding delay for the instruction that is being processed by the HD44780. Figure 3.6 below illustrates the hardware for switching the delays for each of the instructions.

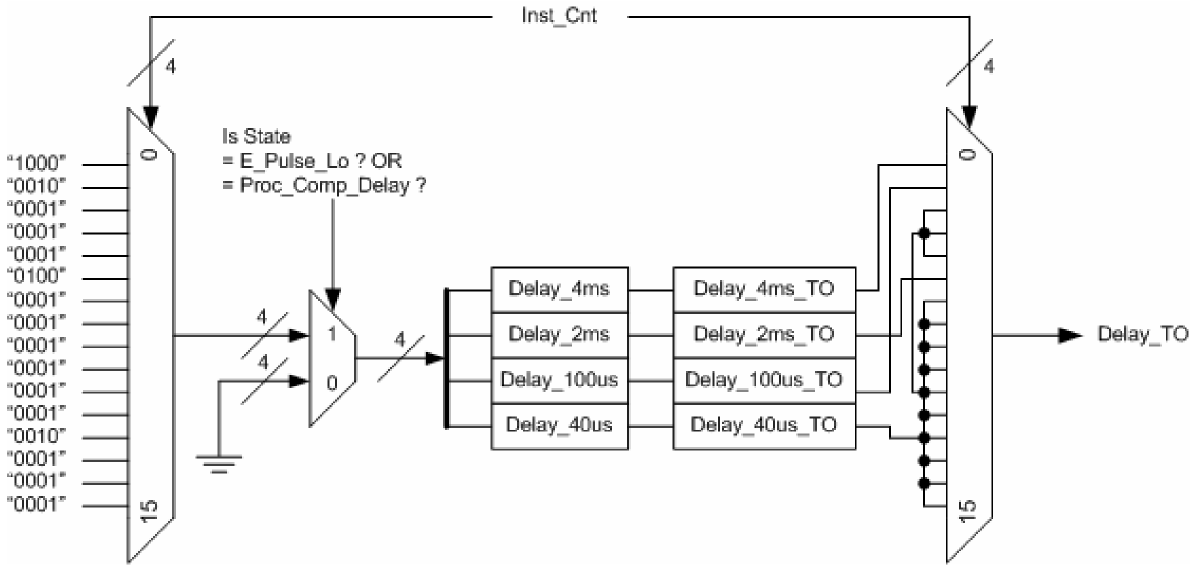

Figure 3.6. A bulk delay controller to select the appropriate delay for the instruction being executed.

The *Inst_Cnt* signal selects the appropriate delay to enable and time its completion in the Time Out (TO) entity as shown in Figure 3.6 above. The TO circuit comprises of a layers of AND gates to ensure that all the output bits of the selected delay element are '1'. Hence, when any of the delay elements reaches the maximum value, TO entity will output logic '1' to indicate the selected delay time has elapsed. When *Delay_TO* is set to '1', the FSM moves on to the next state, *Load_Next_Data*. Otherwise, the FSM stays at the *Proc_Delay_Comp* state. Note that the delay elements here correspond to the actual delay needed by the same sets of instructions as selected by *Inst_Cnt* in Figure 3.1 above.

Referring back to Figure 3.1 above, the CE is set to '1' when the FSM is in the state of *Load_Next_Data*. When CE is asserted high, the counter increments itself and this automatically selects the next instruction to be sent to the HD44780. However, if the Inst_Cnt has reached its maximum value, i.e. 15, the FSM will then move to the final *End_State*. Otherwise, the FSM will continue sending instructions to the HD44780 by making a transition back to the *Write_Data* state.

4. FPGA Implementation

The FSM and hardware layout proposed earlier is implemented on a Spartan-II XCS200-5 FPGA. Both the FSM and hardware layout are synthesized using the Xilinx ISE 8.1i VHDL Compiler. The resulting area occupancy with the maximum setting for the place and route effort is 87 out of 2352 slices. The minimum period for the clock signal applicable to the design is 9.176 ns, which translates to a maximum clock frequency of 108.98 MHz.

5. Field Testing

Running a test bench simulation for this design would be computationally heavy, considering the fact that the delays are much higher than the period of the applied clock frequency (50 MHz). Therefore, simulation is omitted and a field test is directly conducted. The test circuit is implemented on a XESS XSA-200 board which houses the Spartan-II XCS200 FPGA. Figure 5.1 below shows the circuit used to field test the design.

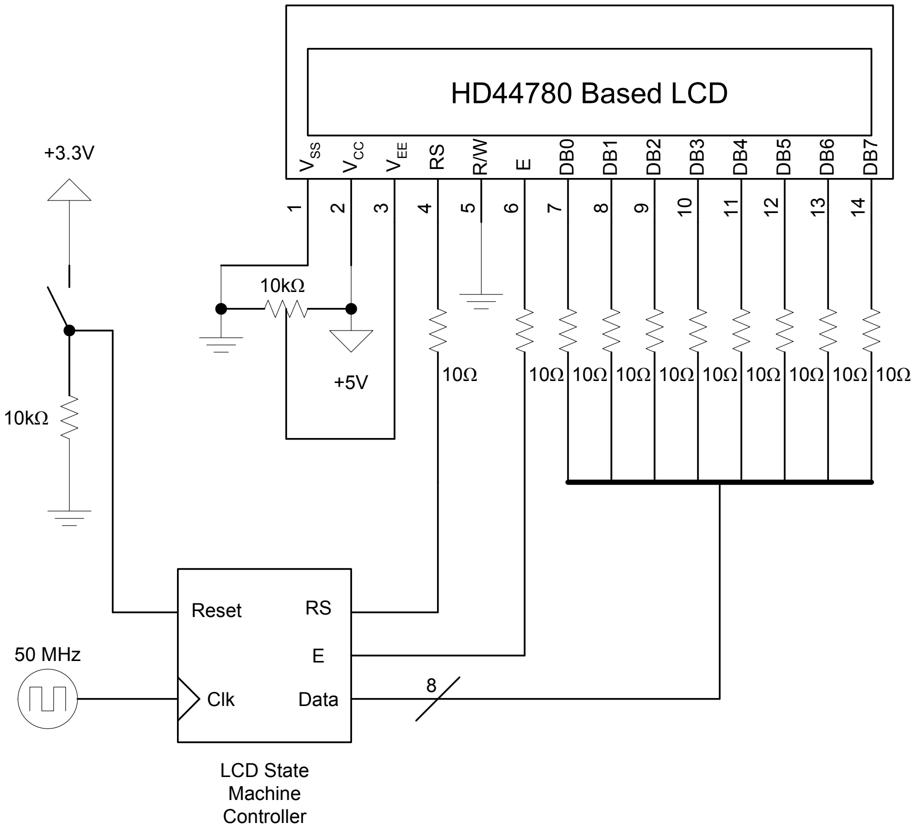

Figure 5.1. Test circuit for performing a field test on the hardware design.

The Reset pin is pulled low by a 10kΩ pull-down resistor. When the button is pushed, the LCD State Machine Controller is placed in a Reset state. A 50 MHz clock input is fed into the FPGA clock pin to drive the LCD State Machine Controller. The RS, E and Data pins are connected accordingly to the HD44780 LCD via a 10Ω resistor to prevent the FPGA from sourcing out too much current. Finally, a potentiometer is connected to the V_{EE} pin between V_{CC} and V_{SS} so the level of contrast of the LCD may be adjusted. Figure 5.2 below shows the circuit in Figure 5.1 being field tested.

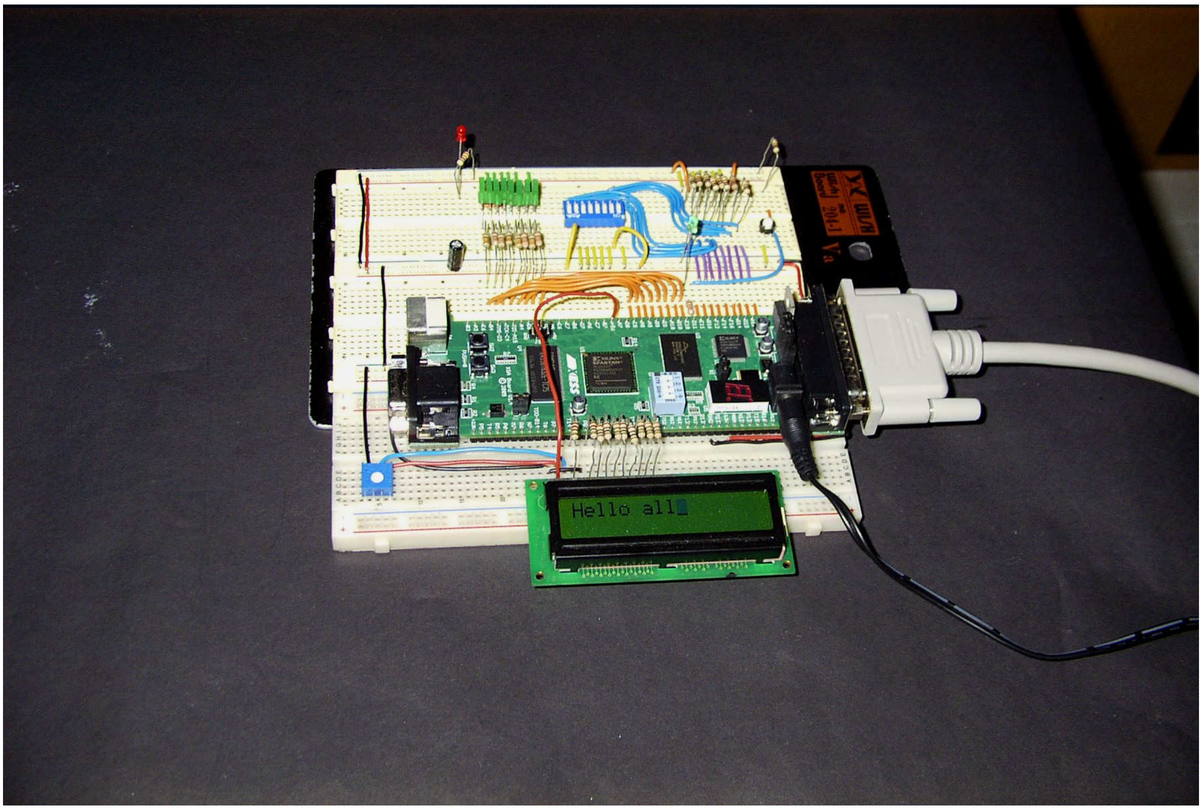

Figure 5.2. Testing the controller with an actual HD44780 based Text LCD.

6. Conclusion

A simple and compact way of controlling the HD44780 based Text LCD is presented and its delayed Finite State Machine (FSM) is explained. The delayed FSM can too be modified to accommodate more complex designs, simply by modifying the size of the multiplexer to one that can select more instructions, for e.g. using a 32 to 1 multiplexer for instance, in addition of increasing the data width of the *Inst_Cnt* count-up counter. Also, modifications can be easily made to the bulk delay controller by expanding the multiplexer to accommodate for more HD44780 instructions.

References

[1] Hitachi HD44780U (LCD-II) Dot Matrix Liquid Crystal Display Controller/Driver Datasheet, Revision 0.0. Hitachi Ltd.

[2] Spartan-3A/3AN Starter Kit Board User Guide. UG330 (v1.2), April 15 2007. Xilinx. http://www.xilinx.com

[3] www.fpga4fun.com