# Scrolling Message on 7-segment displays

List of Authors: Grant Mckee, Jason Wegryn, Zachary Kilmer
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: jwegryn@oakland.edu, grantmckee@oakland.edu, zkilmer@oakland.edu

Abstract- Using the Nexys A7 FPGA trainer board with its 7-segment displays, and VHDL to create a scrolling message across all 8 of the displays. This report will cover the design and implementation of this system. The purpose was to use all of the knowledge that was taken from ECE 2700 and use it to develop a complex and unique system.The Scrolling 7-segment displays uses different MUX's, a register, multiple FSM's, multiple counters, and a 3-to-8 decoder. Major findings are to build big components like the supershiftreg connected to the FSM's separately to erase some hassle in the implementation later.The major recommendation in order to successfully complete this project is to get the circuit and connections down on paper and understand it before going into the implementation of the VHDL coding.

## I. INTRODUCTION

Our group will be creating a scrolling banner on a 7-Segment display, with the message "ECE 2700". Code for the project will be written in VHDL. The scrolling effect of the letters will be created by using registers to shift the values of each letter to the next display. The user will use the switches to select different scroll speeds, scroll direction, and orientation of the letters on the banner. The scrolling direction feature will be implemented using multiplexers to change the direction at which the intended message will appear on the displays. The scrolling speed will be controlled by counters. Then, the orientation of the message will be controlled by multiple FSM's that will be selected by the user using switches as the select bits of a 4-to-1 MUX. The full connections and block diagram of the circuit is shown in Figure 3 below. The motivation for this project is to use aspects of what we learned in the classroom and implement them in a real life application. The class material that will be used in this project will be registers, multiplexers, FSM's, counters, 3-to-8 decoder, and 7 segment displays. All of these components that were introduced in the class will be key components in the implementation of our design. The implications of our project will be substantial as we use what we learned in class to design and implement a project. Using the information we know and having more exposure to it will allow us as engineers to progress our knowledge in the field as well as further ourselves by having to search for more information about a particular topic. These things will allow us to be better engineers in the future. The applications of this project in the real world are limited but it will test us in almost everything that we learned in the course. Some possible applications would be for advertising. One that you might see often are the electronic billboards all along the highway. The simple ones have a sliding message that sometimes will have a different scrolling speed depending on importance.

## II. METHODOLOGY

### *A. Counters*

The counters in our design will be used in order to control the enables of various items in the implementation. The first counters, which

will control the enable of FSMreg, FSMregleft, FSMregleftupsidedown, FSMregrightupsidedown, and shift-registers, are labeled as Counter 0, Counter 1, Counter 2, and Counter 3. These counters will be used to control the scrolling speed of the message across the displays. For example, Counter 0 will release a pulse every 1.5 secs this will cause all of the FSM's connected to change states. This will also cause the shift-registers to shift every 1.5 secs, which is important as this is the component that is scrolling the message. This counter will keep these two components in sync which will protect the system from potential glitches. The other counter in the system will control the enable of the FSM. This counter will generate a pulse every 0.001 secs, which in turn will cause the state of the FSM to change every 0.001 seconds. Which is needed for our project because the 7-segment display has 8 displays all connected to one connection so we need to be able to have the FSM change its state quickly so that the MUX and 3-to-8 decoder also change very quickly. If this was not changing as fast as we have it, a person would be able to see each letter displayed at once, because the 7-segment display can only display one letter at a time, but with this implementation we are able to display each character so quickly that the human eye cannot detect it. Thus, making it look to us as a solid message.

*B. FSMreg's*

The FSMreg will control what will be inputted into the shift-register based on the state of the enable as well as the current state it is in at that time. The state change time will be controlled by counters in the system. As seen in the ASM table below, states are laid out inorder to display this message from right to left. The FSMreg will start in State 1 and when the enable is '1' then the FSMreg will progress to the next state. When this happens the FSMreg will output 'E', already decoded for the 7-segment display. This output will be transferred to the input of the shift-register where it will be shifted in order to give the "scrolling" effect to our message. This can be observed in Figure 1 shown below. The FSMregleft, FSMregleftupsidedown, and FSMregrightupside down will follow the same ASM diagram other than the change of what is being outputted during each state. This system will continue to progress through the states as shown in its particular ASM diagram shown below. Then, once the last state is satisfied it will go back to state 1 and then repeat the process over again. This system will input our message into the shift-register. These same principles will work for the other 3 FSM's labeled, FSMregleft, FSMregleftupsidedown, FSMregrightupsidedown. These different FSM's will just change the message as depicted in the name. The FSMregleft will be the correct orientation when shifting the message from left to right. FSMregleft will also be for when you are shifting from left to right but the message will be upside down. FSMregrightupsidedown will be for when the message is shifting from right to left but the message will be inverted. Using these different FSM's, I was able to make the display readable when shifting from right to left and left to right as well as rotate the message so it will be upside down while crossing the displays.

*C. FSM*

The FSM will control the select bits for the 8-to-1 MUX's, as well as, send those select bits as an input on the decoder. These tasks will cause the FSM to work as a controller for the decoder as well as the 8-to1 MUX's. The importance of this controller is to ensure that based on what select bit is being sent that the correct displays will be on/off depending on the progress of the scrolling message. The FSM works as shown in the ASM diagram shown in Figure 2. When the 0.001 sec counter sends a '1' onto the enable of the FSM, then the system

will progress to the next state and output a particular select bit according to Figure 2 below. This system will progress as follows until it reaches the last state. When the last state is satisfied the FSM will return to state 1 and repeat the process. This is how our FSM component will work in our design.

*D. 3-to-8 Decoder*

The 3-to-8 decoder will receive a 3 bit signal from the FSM. This 3 bit signal will determine what displays will be on as well as what displays will be off. These select bits will be imputed every 0.001 secs which means that the displays will be turning on and off so fast that it will look like a solid message to the human eye. This is also a key component to create the "scrolling" effect on our message.

*E. 4-to-1 MUX's*

One of the 4-to-1 MUX will be the controller of what clock the enable of the FSMreg, FSMregleft, FSMregleftupsidedown, FSMregrightupsidedown will follow. The inputs of the MUX will be the four counters and the output of the MUX will be one of the clock signals based on the select bits being inputted by the user. The user will input the select by flipping two switches on the NEXYS Board. The switch values will work as a 2 bit signal that goes into the MUX and selects which clock will be used to scroll the message across the displays. This 4-to-1 MUX will act as the speed controller for the scrolling of our message. The other 4-to-1 MUX will control which of the FSMreg's go into the input of the shift register. The inputs of the MUX will be the 4 different FSMreg's and the output will be the FSMreg chosen by the user. The user will be able to choose which FSMreg by flipping 2 switches on the NEXYS board.

*F. 8-to-1 MUX's*

One of the 8-to-1 MUX will be used to select what output from the 8 shift register will be displayed on the 7-segment display. This was pivotal to the success of our project as the 7-segment displays all run off 1 connection. This MUX will select the letter that we want to display based on the shifting process or the shift register, as well as, the select bit. This MUX will select a new letter every 0.001 secs because the system is independent of the clock and it will be presented with a new select bit every 0.001 secs. This is what causes it to choose a new position so quickly. This works in sequence with 3-to-8 decoder so that when the letter is picked to be put on the 7 segment display it can know exactly where to go by what display is on according to the 3-to-8 decoder. The top 8-to-1 MUX will be used to shift the message from right to left across the 7-segment displays. Where the 8-to-1 MUX below that will be used to shift the message from left to right across the displays.

*G. 2-to-1 MUX*

The 2-to-1 MUX will be used to control which way the message is scrolling. This MUX will allow the message to scroll from right to left when the user puts the select bit to 0. Then, when the user changes the select bit to 1 then the message will change directions and start scrolling from left to right. This will allow the user to rotate the message as well as to shift the message in either direction and orientation when working in accordance with the 4-to-1 MUX that controls what is going into the shift register.

*H. Shift Register*

The shift register (aka super shift register) component of our design is actually made up for 8 registers that will all output the current

letter on Q, as well as, allow that Q to the input on the next register so that during the next clock cycle the message can continuously shift. As the chosen FSMreg keeps feeding the lead register new letters the rest of the registers are shifting the information that was collected from the lead register to the next register and so on and so forth. This is key to the scrolling of the message across the displays as while this is shifting it is changing the inputs on the 8-to-1 MUX which allows the MUX to output the message at different points as it travels from the far right of the display to the far left of the display. The rate at which it shifts will be dependent on the enable it is receiving from one of the four counters chosen by the user, as stated previously. This allows the user control scrolling speeds. This is the motor of our whole design and is the most complex component of our system.

## III. EXPERIMENTAL SETUP

Using Vivado and VHDL we were able to create, test, and simulate our project design and upload it to the Nexys A7 board. These devices and programs were used inorder to give our project life. Vivado was the program provided to us and which enabled the use of the VHDL hardware coding language which was needed to program our NEXYS DDR4 board. The behavioral simulations in Vivado helped us to test the board as well as tweak errors on a signal by signal basis instead of having to guess by testing it exclusively on the NEXYS DDR4 board. The expected results were to see the characters shifting in the shift register as well as on the outputs of the 8-to-1 MUX's. These things were key observations because once these items were observed it was known that the shifting of the message was working as it should. Vivado is a very well versed and useful program for us as engineers to use in order to code in VHDL. The tools that are in place to help are useful and can make debugging simple.

## IV. RESULTS

Based on the circuit design, the message "ECE 2700" was properly displayed across the seven segment displays. Using the switches on the board varying scrolling speed, direction, and orientation of the message. This was the expected result of our project. The key findings that were found during the implementation of this program was that each program can be done a multitude of different ways and that no one way may be the correct way to do it. This directly contributes to what was learned in class as our professeur had explained to students that it can be done multiple ways and possibly easier ways but he wanted us to do it the long way first to learn. The other findings in the creation of our project is that all MUX's are extremely useful for allowing the user to make decisions in the circuit in order to change the outcome at the end. This also related to our class as we learned how multiplexers are used in lots of circuits and can give options as we solved boolean equations using Shannon's expansion to get the circuit of that function. Lastly, it was observed that counters, registers, and FSM are useful in a wide range of applications and it shows as seen in our implementation, we implemented many of these items into our design. These observations were expected as it was known that a Scrolling 7-segment display would test our skills in a multitude of items learned throughout the semester. These occurrences observed during the implementation could all be explained as we have run into these many times in the lab as well as the classroom setting.

## CONCLUSIONS

Some of the main take-away from this project is learning how to debug any circuit as well as develop the project on paper before going head first into something that may not be fully conceptualized. This is a very important tactic to have when dealing with any problem as well as

dealing with problems in coding or designing a system. The debugging of code will be useful in our careers because some of us are computer engineers and will need this ability to debug efficiently more than ever. There is still a need for all engineers to know how to code and be able to debug circuits, as all industries move toward a more A.I. approached workplace. Some issues that were encountered when designing this system were the 8 shift registers working in unison and how they would shift the bits. The top file also caused some issues as to how each component was connected and how it would affect the system as whole. These challenges were met and were fixed inorder to ensure the project worked as it was expected. Some improvements that could be made are giving it the ability to rotate a single character so it could have a more dynamic look as it crossed the display. Another improvement would be to use a LCD display instead of a 7-segment display. The reason for this is that it would allow the user to use any message he or she wanted as you would not be restricted to certain characters on the LCD display as you are on the 7-segment display. These improvements to the project would give it much more application in the real world today.

References

[1] D. Llamocca,"VHDL Coding for FPGAs." Accessed November 24, 2020. https://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html.

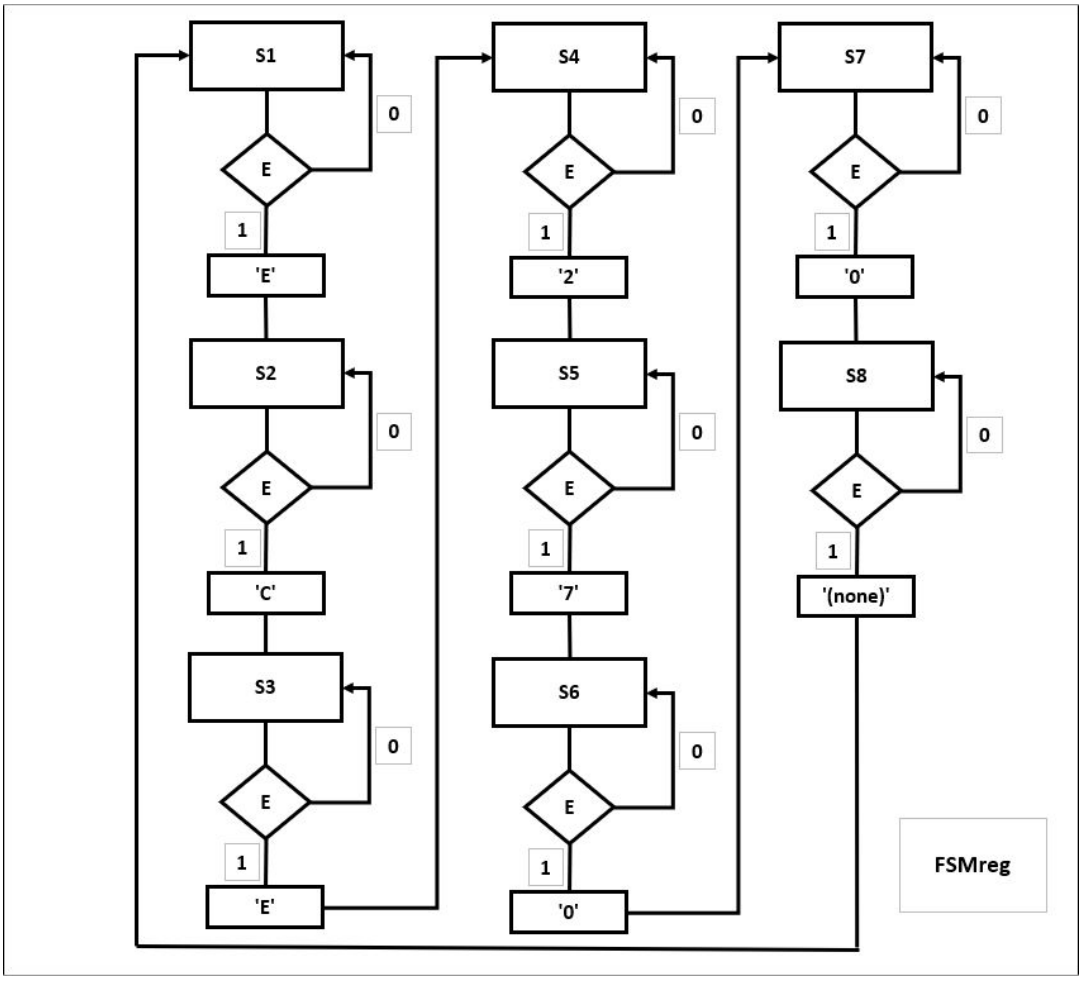


**Figure 1. FSMreg ASM Diagram**

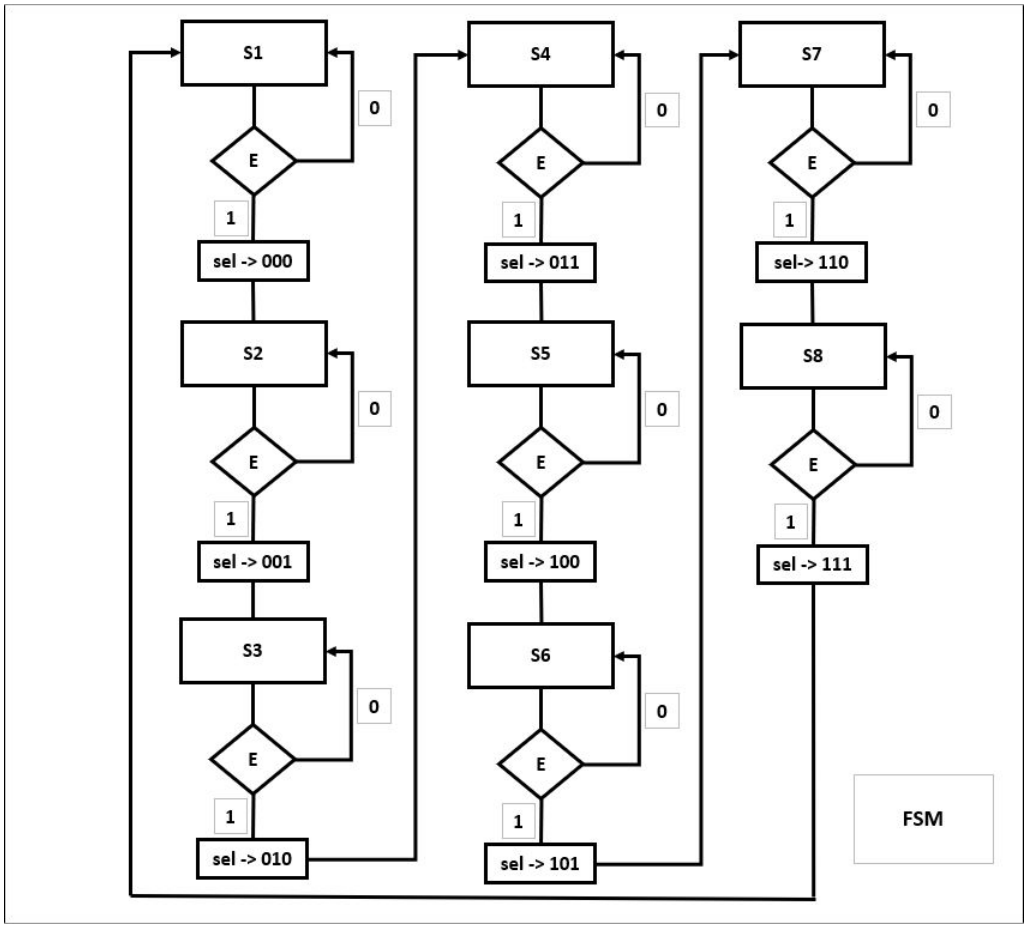


**Figure 2. FSM ASM Diagram**

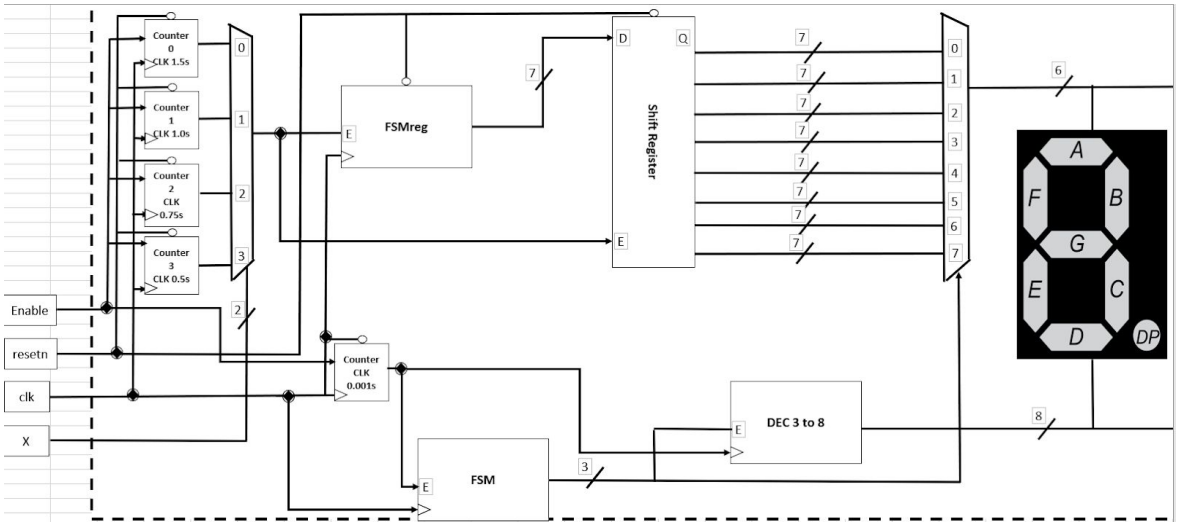


**Figure 3. Block Diagram**