	ECE410-Advanced Digital Logic Design
	LAB 2: Sequential Circuits and Finite State Machines Using VHDL.
Fall 2024	

DATES

Section	Date	Time	Place
D11	Session 1 07-Oct-2024 Session 2 21-Oct-2024	2:00 to 4:50 PM	ETLC E5-001
D31	Session 1 09-Oct-2024 Session 2 23-Oct-2024	2:00 to 4:50 PM	ETLC E5-001
D51	Session 1 11-Oct-2024 Session 2 25-Oct-2024	2:00 to 4:50 PM	ETLC E5-001

INTRODUCTION

Sequential circuits form the foundation of many advanced digital systems. Unlike combinational circuits, sequential circuits rely on memory elements and feedback to maintain state information, making them essential for tasks that require state-based behavior. Key examples of sequential circuits, such as shift registers and linear-feedback shift registers (LFSRs), are used extensively in data storage, manipulation, and pseudo-random number generation. One important application of sequential circuits is the Finite State Machine (FSM). FSMs are the building blocks of both simple controllers and advanced digital systems and are used to solve many key problems in computer science, such as search algorithms, pattern matching, and artificial intelligence. In this lab, students will design and implement sequential circuits and various types of FSMs using both behavioral and structural approaches.

LEARNING OBJECTIVES

- To gain familiarity with behavioral-level VHDL design through the implementation of a linear feedback shift register.
- To design finite state machines (FSMs) using VHDL, with a focus on robust design methodologies and coding best practices.
- To perform functional verification of FSMs by designing and executing testbenches that rigorously evaluate system behavior.
- To implement an automatic door controller as well as emulating its inputs.



ECE410-Advanced Digital Logic Design

LAB 2: Sequential Circuits and Finite State Machines Using VHDL.

Fall 2024

PRE-LAB

Marks 10%

- Carefully read this document and follow the detailed instructions below.
- Draw a block diagram of an 8-bit Galois LFSR with taps on positions 1, 2, 3 and 7.
- Draw a block diagram of an 8-bit Galois LFSR with taps on custom positions that you choose.
- Draw a state diagram for each count mode of the count generator from Part 2.
- Draw the state diagram representing a Finite State Machine (FSM) for the door controller, clearly illustrating the transitions between states along with their input and output conditions.
- Prepare a list of five real world applications that use state machines at their core. Do not include the examples from the first page of this handout.

Note: 10% of the Lab 1 mark will be allocated for the pre-lab work. Skipping the pre-lab work may prevent you from finishing the exercises within the 3-hour lab time. You are to submit your pre-lab work as a separate PDF file individually on the eClass site at least one hour before coming to your lab session.

Note: You must demonstrate your working designs to a TA or LI for all parts of this lab.



PART 1: PSEUDO RANDOM NUMBER GENERATION AND LFSRs

Marks 10%

Resources needed:

- ➔ *display_driver.vhd*
- ➔ *Zybo-Z7-Master.xdc*

Shift registers are widely used in digital systems for different kinds of operations. They consist of a linear array of flip-flops, where the output of one flip-flop feeds into the input of the next, enabling data to shift through the register at each clock pulse. Shift registers can operate in different modes depending on how the data is loaded and retrieved. They are crucial in applications like data buffering, signal processing, and in the generation of pseudo-random numbers using Linear Feedback Shift Registers (LFSRs).

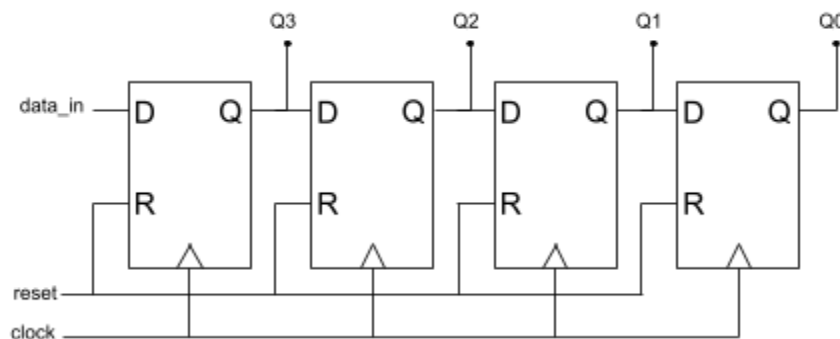


Figure 1: 4-bit Shift Register Block Diagram

LFSRs build upon basic shift registers by introducing a feedback mechanism that enables the generation of pseudo-random sequences. There are two main types of LFSRs: **Fibonacci** and **Galois**. While both use a series of XOR operations on the flip-flop outputs to create feedback, they differ in the placement of the feedback circuit and hence the shifting behavior:

- In a **Fibonacci LFSR**, the feedback is applied after the shift operation, with specific taps selected to XOR certain bits that feed into the input of the register. The taps are placed at certain flip-flop outputs
- In a **Galois LFSR**, the feedback is applied directly in between some of the register stages during the shifting process, which allows for more efficient implementation and often leads to faster hardware designs.

LFSRs are commonly used in cryptography, digital communications, and error detection because of their ability to generate long sequences of pseudo-random numbers with minimal hardware. Pseudo-random numbers are generated using a deterministic state machine but they have statistical properties similar to those of truly random



ECE410-Advanced Digital Logic Design

LAB 2: Sequential Circuits and Finite State Machines Using VHDL.

Fall 2024

numbers. By leveraging specific feedback tap configurations, LFSRs can cycle through a deterministic sequence of values that approximates random behavior.

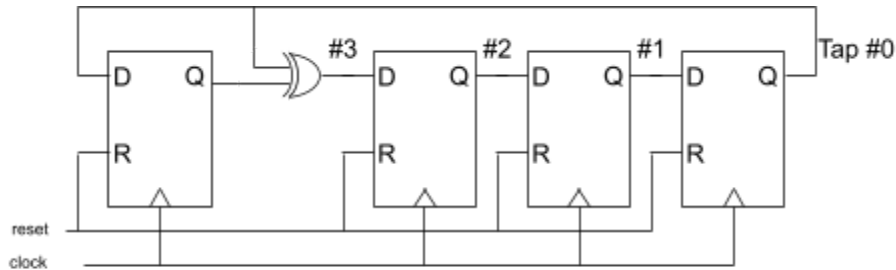


Figure 2: 4-bit Galois LFSR with taps on 0 and 3

For this part of the lab, you will design an 8-bit Linear Feedback Shift Register (LFSR). You will implement two different linear feedback connections with different sets of feedback taps. You will then compare the output sequence produced by the two LFSRs.

First, design both LFSRs by writing a behavioral VHDL model for each configuration. The first configuration will have taps at positions 1, 2, 3 and 7, while the second configuration will involve a different set of taps (you are free to choose your own configuration). Note the convention for tap numbering in Galois LFSRs that is used in Figure 2.

Next, write a testbench that can be used with both configurations to verify their functionality. This testbench should simulate the behavior of the LFSRs, allowing you to compare the sequences generated by both configurations. You will observe differences in sequence length, randomness, and repeating patterns. Document your observations carefully and ensure you identify any differences between the two designs. Pay attention to the maximum length of the pseudo-random sequences before repetition produced at tap #0 by each configuration and analyze their overall performance. In your report you must describe how you measured the degree of randomness of the bits output at tap #0. Reflect on the significance of these variations in real-world applications where LFSRs are used.

For the last step of this section, you will implement your LFSR on the Zybo Z7-10 board. Follow these steps:

1. **Asynchronous Reset:** Add an asynchronous reset signal to your LFSR design. This reset signal will allow you to clear the LFSR and start over from the initial state. Map this reset signal to one of the pushbuttons on the Zybo board.
2. **Visual feedback:** Map the output vector of the LFSR to a 7-segment display using the `display_driver.vhd` provided in the resources.



ECE410-Advanced Digital Logic Design

LAB 2: Sequential Circuits and Finite State Machines Using VHDL.

Fall 2024

3. **Clock Signal Configuration:** Use a 2 Hz clock signal to drive the LFSR. You can generate this clock by dividing the board's system clock using a clock divider. This clock speed will allow you to visually observe the changing values on the seven-segment display.
4. **Manual Constraints File Editing:** You will need to add the pin constraints manually. Modify the **Zybo-Z7-Master.xdc** file to map the required signals to the appropriate physical pins on the Zybo board.



ECE410-Advanced Digital Logic Design

LAB 2: Sequential Circuits and Finite State Machines Using VHDL.

Fall 2024

PART 2: FSM COUNTER WITH CUSTOM SEQUENCE

Marks 15%

For Part 2 students will design and implement a finite state machine for a count sequence generator based on its state transition diagram. The FSM will generate 4-bit sequences and allow the selection of different counting modes through a select line.

Begin by creating a new Vivado project and a design file named **count_generator_fsm.vhd**. Using the state diagram created in the prelab as a guide, write a VHDL model to describe the FSM behavior. You must decide whether to implement either a Mealy or Moore type FSM. Justify your choice based on the design requirements and expected behavior of the count generator.

The count generator should have two modes controlled by a select line:

- **0** selects an up-counter
- **1** selects a custom sequence designed by the students.

The FSM should output a 4-bit sequence corresponding to the selected mode and it should ensure correct transitions between the states based on the select line and the clock input.

Once your VHDL model is complete, create a testbench to simulate the FSM's operation and test all possible input conditions (e.g., different select line values and sequence behaviors). Ensure the testbench includes assertions for each state transition to verify the correct behavior of the count generator in all modes. For example: **assert not (A='1' and B='0') report "Either A='0' or B='1'";**

For the custom sequence, you are required to design a 4-bit count sequence in which exactly three bits change when transitioning from one state to the next. Each state transition should be well-defined. Ensure that the sequence starts at **0000** and ends at **0000**, completing a full cycle while maintaining the three-bit change rule at every transition.

For the last step, you will now implement your FSM design on the Zybo Z7-10 board. Follow these steps:

1. **Asynchronous Reset and Enable (EN) Inputs:** Add an asynchronous reset input to your FSM to clear the current count and go to the reset count state. Additionally, provide an enable (**EN**) input to control whether the FSM continues counting or pauses at its current state. Map the reset signal to a pushbutton and the EN input to a switch on the Zybo board.
2. **Input Mapping:** Map the select line (**0** for up-counter, **1** for custom sequence) to a switch of your choosing on the Zybo board. This will allow you to change the counting mode dynamically.
3. **Map the Output to the 7-Segment Display:** Similar to Part 1, map the 4-bit output of your count generator FSM to a seven-segment display using the **display_driver.vhd** provided in the resources.



ECE410-Advanced Digital Logic Design

LAB 2: Sequential Circuits and Finite State Machines Using VHDL.

Fall 2024

To adapt it for this part, you will need to modify the `display_driver` file to display only one character. **Tip:** This modification will involve adjusting the `display_select` output logic.

4. **Clock Signal Configuration:** Similar to Part 1, use a 2-Hz clock signal to drive the FSM.



PART 3: AUTOMATIC DOOR CONTROL SYSTEM

Marks 20%

In this section, you will design and implement a finite state machine for an automatic door controller based on its state transition diagram. The door controller will manage the door's four states: DOOR_CLOSED, DOOR_OPENING, DOOR_OPEN, and DOOR_CLOSING, and control transitions based on two limit switches (for detecting the fully open or fully closed door positions), and a motion sensor that senses movement near the door.

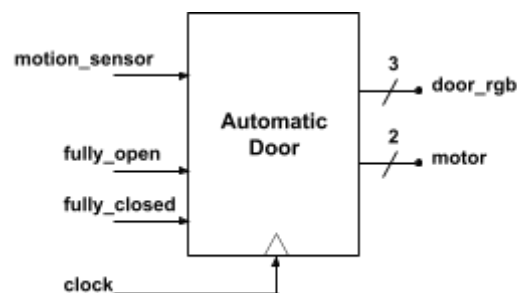


Figure 3: Automatic Door Block Diagram

Inputs:

- **clock:** The system clock signal that regulates the timing and synchronization of the finite state machine (FSM). It ensures that state transitions and other operations occur at consistent intervals, keeping the FSM synchronized with an internal timer.
- **motion_sensor:** A trigger signal from a motion sensor that indicates when movement is detected near the door. This input initiates the opening of the door by signaling the FSM to transition from the closed state to the opening state.
- **fully_open:** A signal controlled by a limit switch placed at the fully open position of the door. When the door reaches this position, the switch is activated, sending a signal to the FSM to indicate that the door is fully open. This triggers the transition from the opening state to the open state.
- **fully_closed:** A signal controlled by a limit switch placed at the fully closed position of the door. When the door reaches this position, the switch sends a signal to the FSM indicating that the door is fully closed, prompting the FSM to transition from the closing state to the closed state.

Outputs:

- **door_rgb:** A 3-bit RGB LED signal representing the current state of the door. The LED should display different colors for each state:
 - **DOOR_CLOSED:** Red
 - **DOOR_OPENING:** Blue
 - **DOOR_OPEN:** Green
 - **DOOR_CLOSING:** Yellow

- **motor**: a 2-bit signal representing the state of the motor as follows: **00** for off, **01** for opening, **10** for closing, and **11** as a don't-care state..

The door FSM should operate as follows:

When the FSM is in the **DOOR_CLOSED** state, it should wait for a signal controlled by the motion sensor. Upon receiving the **motion_sensor** signal, the FSM should transition to the **DOOR_OPENING** state. The door should remain in the **DOOR_OPENING** state until the **fully_open** limit switch is contacted. Then the FSM should transition to the **DOOR_OPEN** state. When the FSM is in the **DOOR_OPEN** state, it should start a timer and wait for a predefined time delay or until the **motion_sensor** signal is deasserted. Once the time delay has elapsed and the **motion_sensor** signal is deasserted, the FSM should transition to the **DOOR_CLOSING** state. The door should again remain in the **DOOR_CLOSING** state, until the **full_closed** limit switch is contacted. If the **motion_sensor** signal is asserted before reaching the **fully_closed** limit switch, the FSM should transition back to the **DOOR_OPENING** state. Otherwise, the FSM should transition to the **DOOR_CLOSED** state.

First, create a new Vivado project and create a design file named **door_controller.vhd**. Using the state diagram created in the prelab as a guide, write the VHDL model to describe the FSM behavior. Ensure that the door controller transitions between the four states based on the signals from the motion sensor and from the limit switches which indicate when the door is fully open or fully closed and the internal timer.

Once your VHDL model is finished, create a testbench file called **door_controller_tb.vhd** Use this testbench to simulate the FSM's operation and test all possible input conditions (e.g., limit switch signals, door sensor). Ensure the testbench that includes assertions for each state transition to verify the correct behavior of the door controller.

Implement your design on the Zybo Z7 board by mapping the inputs (**motion_sensor**, **fully_open**, and **fully_closed**) to interact with the Analog Discovery 2 (AD2). Refer to Table 1 below for the specific physical connections from the AD2 to pins on the Zybo Z7-10.

Output Name	AD2 GPIO	Zybo Z7-10 Package Pin
motion_sensor	0	V15
fully_closed	1	W15
fully_open	2	T11
clock	-	K17

Table 1: Input Signal Mapping

The outputs (**door_rgb** and **motor**) should be mapped as shown in Table 2 below.



ECE410-Advanced Digital Logic Design


LAB 2: Sequential Circuits and Finite State Machines Using VHDL.

Fall 2024

Output Name	Zybo Z7-10 Package Pin
door_rgb[2:0]	V16, F17, M17
motor[1:0]	M14, M15

Table 2: Output signal mapping

Connect the AD2 to the Zybo board using a Pmod cable, and download the generated bitstream to the Zybo board. Connect the AD2 to a PC and open the WaveForms application. Navigate to the **Script** tab and open the `door_test.js` script provided in the lab resources. Then, open the **Static I/O** instrument and run the script. On the bottom of the **Script** instrument tab, you will see console messages indicating the state of the script. Demonstrate your working system to the Lab Instructor (LI) or Teaching Assistant (TA) before proceeding to the last part.

 Fall 2024	<h1 style="text-align: center; color: blue;">ECE410-Advanced Digital Logic Design</h1>
	<h2 style="text-align: center; color: green;">LAB 2: Sequential Circuits and Finite State Machines Using VHDL.</h2>

PART 4: AUTOMATIC DOOR CONTROL SYSTEM UPGRADE

Marks 15%


For this part, you are tasked with implementing an upgrade to your automatic door control system to make it more suitable for real-world implementation. This upgrade will add a new state to the FSM called **EMERGENCY** and introduce new inputs to the door control system. The **EMERGENCY** state will only be reachable under certain conditions, as explained below.

The transition to **EMERGENCY** should happen under two conditions: if the door stays in the **DOOR_OPENING** or **DOOR_CLOSING** state for too long (more than 5 seconds), it may indicate that there is an obstacle blocking the way or that the door has become stuck. In this case, the FSM should transition to the **EMERGENCY** state and remain there until the system is reset. In the **EMERGENCY** state, the motor should be shut down to prevent operation under undesirable conditions, and the RGB LED should flash at a frequency of 1 Hz.

Additionally, you will add a new **force_open** input to the system. This input allows the door to be manually opened regardless of the current state or other conditions. It provides a way to override the automatic operation and force the door open when needed, such as for maintenance or emergency purposes.

Steps:

1. Start by updating the state transition diagram to accommodate the new **EMERGENCY** state, specifying the necessary conditions for transitions to and from the new state from the existing states.
2. Add an asynchronous reset to bring the system back to its initial state. Update the physical constraints by mapping the reset signal to a pushbutton of your choice.
3. Add a **force_open** input to manually open the door regardless of other conditions. Update the physical constraints by mapping the **force_open** input to a switch of your choice on the Zybo board.
4. Modify the logic of your FSM design to include the new state. Verify that the conditions for transitioning to and from the **EMERGENCY** state are correctly implemented.
5. Update the testbench used in the previous part of the lab to include the new conditions for transitioning to and from the **EMERGENCY** state. Simulate various scenarios to verify that the FSM behaves correctly under both normal and emergency conditions.
6. Connect the AD2 to the Zybo board using a Pmod cable, and download the generated bitstream to the Zybo board. Connect the AD2 to your PC and open the WaveForms application. Navigate to the **Script** tab and open the **door_test.js** script provided in the lab resources. Then, open the **Static I/O** instrument and run the script. On the bottom of the **Script** instrument tab, you will see console messages indicating the state of the script. Demonstrate your working system to the Lab Instructor (LI) or Teaching Assistant (TA) before proceeding with the last part.

 Fall 2024	ECE410-Advanced Digital Logic Design
	LAB 2: Sequential Circuits and Finite State Machines Using VHDL.

LAB REPORT

Marks 30%

Submit all simulation test benches, block diagrams and state diagrams, waveforms, RTL elaborated designs, and implementation results in your lab report. Ensure that all of the **.vhd** and **.xdc** files used in the lab are properly identified with your names, following the provided header model. Upload a zip folder containing your report saved as a **.pdf** and all code files. Submit your work using the Lab 2 submission link in eClass before the deadline for your section.

Along with your submission, please address the following questions in your lab report:

1. What was the biggest obstacle you had to overcome while working on this lab, and how did you resolve it?
2. When comparing the two LFSR configurations, what differences in sequence length and randomness did you observe? How did you assess the degree of randomness in the bit sequence? Which configuration produced a longer or more random sequence, and why?
3. Compare the testing process using the AD2 with the behavioral simulation. How did the real-time testing with the AD2 differ from the purely simulated environment, and what are the key benefits of using the AD2 for testing your design? Were there any specific challenges or insights you gained from this hands-on testing that were not apparent during simulation?
4. Suggest other possible upgrades that could be added to the door controller to improve its functionality or safety. Provide a brief explanation for each suggestion.
5. What trade-offs did you consider when deciding whether to use a Mealy or Moore FSM for the door controller? How did the decision impact the complexity and performance of your design?



ECE410-Advanced Digital Logic Design

LAB 2: Sequential Circuits and Finite State Machines Using VHDL.

Fall 2024

References

- [VHDL Essentials YouTube Playlist](#)
- [VHDL Essentials Github Repository](#)
- [Zybo Z7 Reference Manual](#)
- [Analog Discovery 2 Reference Manual](#)
- [Analog Discovery 2 Specifications](#)
- [Analog Discovery 2 Digilent Youtube Playlist](#)
- [PMOD Seven Segment display Reference Manual](#)