Mixed Electronics Lab 8
State Machines

Riley Myers, Caitlin Armstrong

October 1, 2019

1 Introduction

The students will learn to design and state machines that implement part of a communication interface. This will require the students to translate written descriptions of a protocol into a state machine, translate that state machine into Verilog, and test the state machine through both simulation and implementation on their FPGA. They will need to apply their knowledge of Verilog and innovative design skills to fully implement the specified communication system.

2 Field Programmable Gate Arrays

Field-Programmable Gate Arrays (FPGAs) are integrated circuits containing large amounts of programmable logic blocks and RAM, in addition to a hierarchy of interconnects between these blocks. Unlike an application-specific circuit, a FPGA can be reprogrammed, or reconfigured, once it is already in a circuit. As compared to typical microcontrollers, a FPGA tends to be faster and more capable of parallel execution, but also tend to cost more. This places FPGAs roughly between custom silicon and microcontrollers in performance and price, with a wide spectrum of abilities depending on the FPGA in question.
Unlike when you are programming a microcontroller by providing a series of instructions for the defined processor core to execute, 'programming' a FPGA is describing the behavior of blocks and the interconnects between them. You are defining the response of the system to various stimuli, rather than providing a linear flow of instructions. As such, remember that despite the similarities in appearance to C, Verilog is not a sequential programming language. Thus, to effectively utilize the resources provided by FPGAs, you cannot simply write C code.
In this lab, you will need at least one seven-segment display, some hook-up wire, and a FPGA. You may need additional components depending on how you design the system.

1. Does your seven-segment display have current-limiting resistors? Does it need current-limiting resistors?
2. What is the maximum output current from the FPGA pins? What is their maximum tolerated voltage?

3 State Machine Design for Communication Systems

Many complex behaviors can be modeled in terms of state machines. As an example, the state machine for how JTAG behaves is shown in figure 2. As such, being able to design and implement state machines is very useful for designing complex systems and interfacing FPGAs with other complex components. In this lab, we implement a state machine for the communication system described below.

Consider a communication protocol that utilizes a fixed clock rate and three wires: two wires for receiving and transmitting data, and one wire for ground. The communication wires are held high when a transmission is not occurring. To communicate, a device sends a frame (figure 1) that that starts with bringing the appropriate line low for one clock cycle, 8 cycles where the line state corresponds to the data to send (logic high corresponds to a binary 1), and one cycle with a value determined by the parity of the bits in the frame. The line is then returned to the high level, and must be held high for a minimum of two cycles

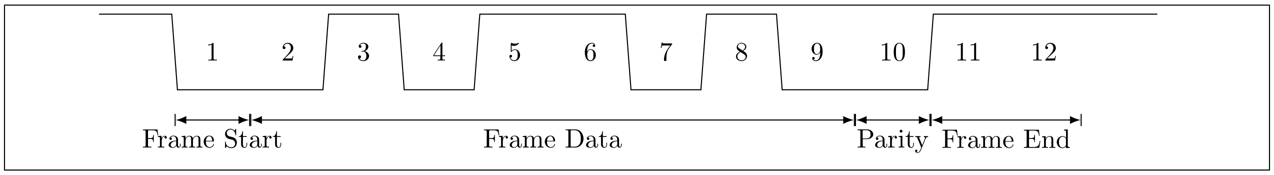

Figure 1: Communication frame containing the value $0x5a$

before the next transmission. After the packet has been received, but before it is considered valid, the parity value must be checked. The parity value is calculated by computing the exclusive-or of all bits in the frame, including the beginning and end framing bits and the parity check. If this value is equal to zero, then the frame is considered valid. If the line drops low for less than a cycle at the start of a frame, the parity value is wrong, or the line is not held high for a minimum of two cycles at the end of a frame, then the frame is considered invalid, so any data received is discarded and an error flag should be set.

For example, one method of implementing this communication system could be a state machine that starts in an idle state, transitions to a state where it checks if the start of a frame is valid, a state (or two) where it samples the bits of the frame, a state where it confirms the end of the frame, and a state where it returns the result. What other states might be useful?

1. Draw a state machine for receiving a message in the communication system described in section 3. What actions should trigger each transition? What actions should occur on each transition or in each state?
2. How would you determine when a frame starts in the communication system? How fast do you need to sample if the clock rate of the communication system is 250 kHz?
3. Design an implementation for the aforementioned state machine on your FPGA. **You do not need to include the parity value verification.**
4. Simulate your design to verify the basic functionality.
5. Design a framework to take the output of the receiver and check if the byte received is an ASCII value. If it is an alphanumeric value, display it on a seven-segment display using the module provided at `https://cs.nmt.edu/~wmyers/labs/segdecode.v`. Otherwise, light up a LED or display some distinct pattern on the seven-segment display to show that an error occurred.
6. Figure out (and implement, ideally) some method to test your design using either external tools or a microcontroller. **Keep in mind that the FPGA inputs have a maximum voltage of 3.4V**

4 Extra Credit

1. Extend the `seven-seg-decode` module to display some of the ASCII symbols, in addition to alphanumeric characters. Some that we think would work are {`[`,`]`,`|`,`!`,`-`,`.`}, but don't let this limit you. Feel free to be creative!
2. Implement the parity bit check and verification. If the frame fails the verification, or if the frame is invalid for any other reason, light up a separate LED.

5 Questions to Consider

1. While this lab highlights some of the benefits of state machines (and other stateful systems), they can have some downsides as well. Consider some of the possible downsides. (Hint: consider their interaction with parallel systems or multiple threads of execution and shared resource)
2. Consider the state machine for sending a message. How similar is it to the receiving state machine?
3. If you feel like you haven't been challenged enough so far (or have extra time), work on an implementation for sending messages in this system.

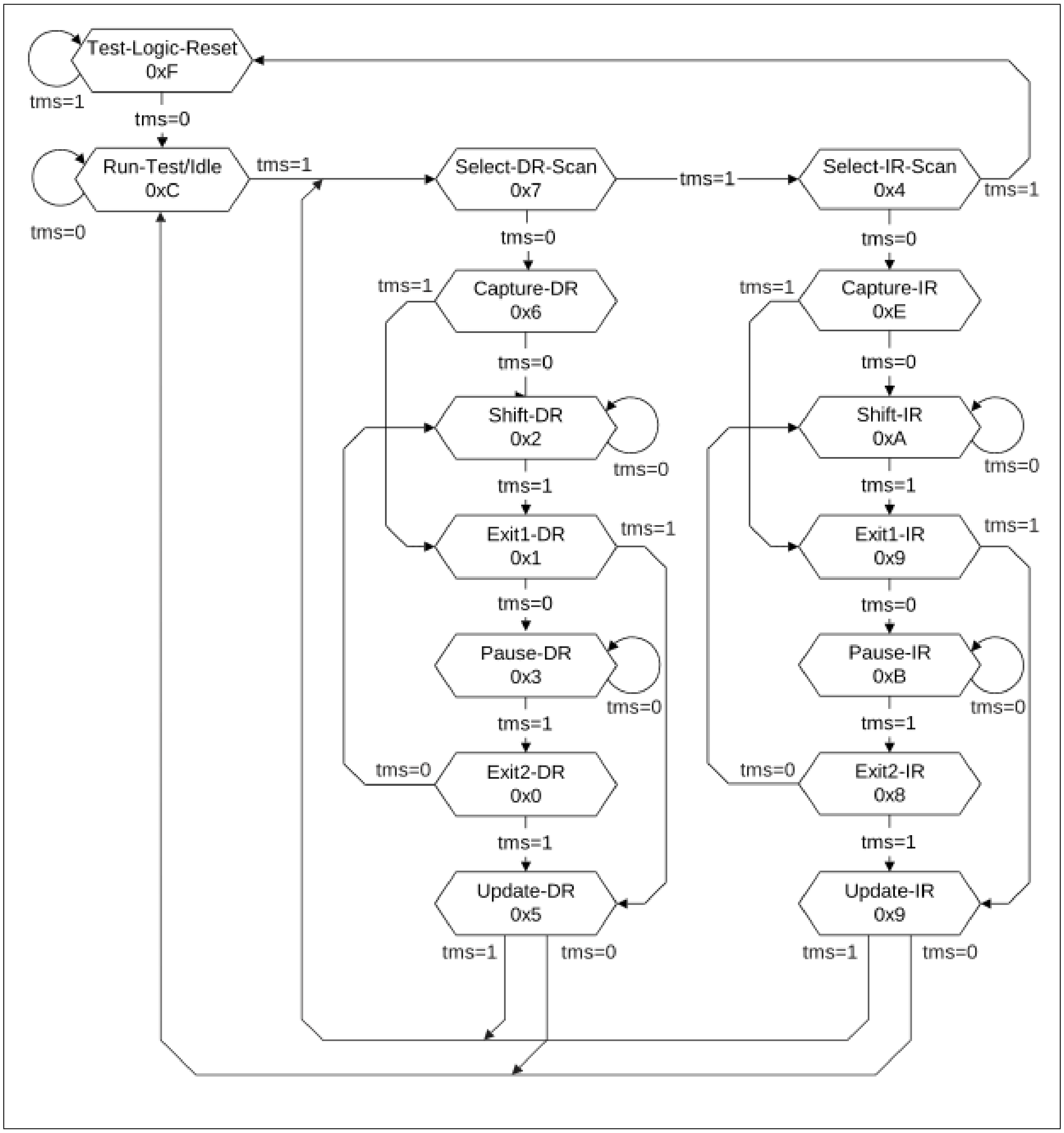

Figure 2: JTAG state machine diagram, shown as an example