EE365 – Advanced Digital Circuit Design
Fall 2003
Design Project 4 – Synchronous Serial Data Transmitter/Receiver

Project Overview:

Your task is to design a system that is able to transmit and receive data in a synchronous manner. The data will be presented to your system in 8-bit words; however, your system is required to transmit them in a serial manner. The serial transmission line will also be used to receive incoming data, and your system will have to output the data as 8 bits in parallel.

Project Background and Design:

Shift registers are often used in parallel-to-serial and serial-to-parallel conversions. A shift register is a series of flip-flops where the Q pin of each flip-flop is connected to the D pin of the next flip-flop in the series. A multiplexer is placed in front of each D pin in order to provide the capability to load the entire register chain with provided data. A functional logic diagram of a shift register is below.

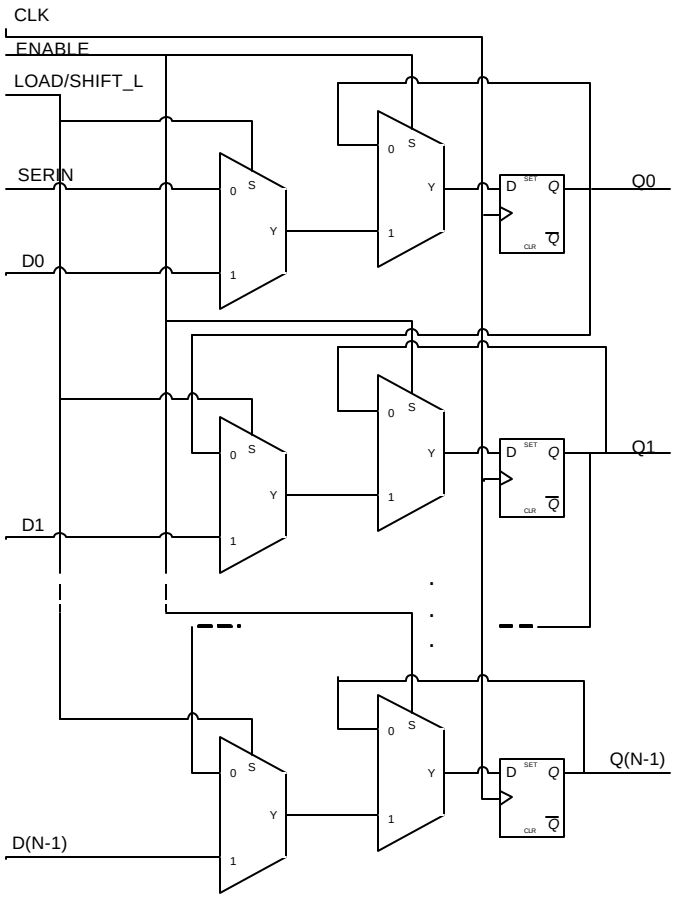

Figure 1: Logic diagram for N-bit shift register.

For the parallel-to-serial conversion, the data is loaded into the register chain by making

the LOAD/SHIFT_L input high. Once LOAD/SHIFT_L is lowered, the data starts shifting up the register chain. The Q output of the last register in the chain is used as the serial output.

For the serial-to-parallel conversion, the register chain is cleared before use. Once the data transmission starts, the data is used as an input to the first register in the chain, and clocked into the chain one bit at a time. Care must be taken to disable the shift register from inputting and shifting data once the last bit of the 8-bit word is received. For more information on shift registers and how they are used, see Wakerly's discussion of shift registers starting on p. 712.

Your system is required to have the following inputs and outputs:

- CLK (input), the clock signal for the system.
- START (input), a signal that is asserted by an external system to latch in the parallel input data and begin serial data transmission.
- STOP_SEND (input), a signal that is asserted by another system to which you are sending data, signaling that it has received the last bit.
- RECV (input), a signal that is asserted by another system sending data to you, signaling that it is about to begin sending data.
- SEND (output), a signal that you send to another system signaling that you are about to begin sending data.
- FINISH (output), a signal that you send to another system signaling that you have received the last bit of data.
- DATA (bidirectional), the serial data line on which you transmit and receive data.
- INPUT_DATA (input vector 7 downto 0), these signals are used to load data to be transmitted.
- OUTPUT_DATA (output vector 7 downto 0), these signals latch received data upon the completion of the receiving process.

Note that the DATA line is bidirectional; in VHDL you can declare a bidirectional data line using the "inout" specifier in your entity declaration. You will also need to create an tristate output buffer to separate the output line when transmitting from the input line when receiving. A diagram of how to use the output buffer as part of the design is below.

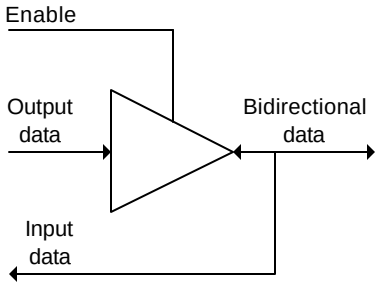

Figure 2: Diagram for usage of a tristate output buffer.

Project Specifications:

Your task is to design a system that uses the inputs and outputs outlined above (as named, more or this later) to implement a system that is able to send and receive parallel data through a serial process. The sending process shall operate as specified by the following timing diagram:

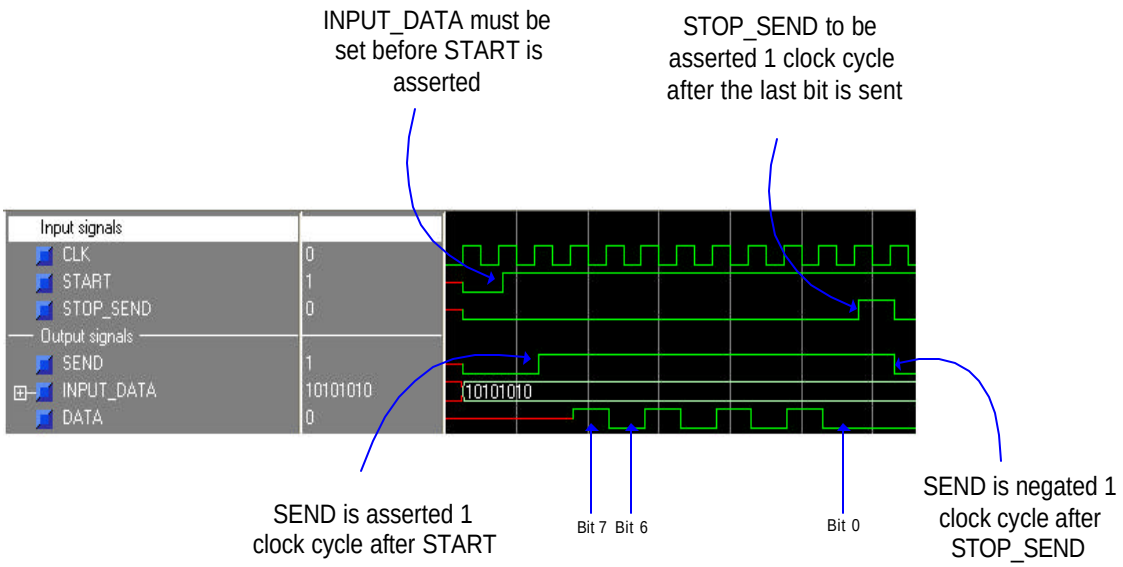

Figure 3: Timing diagram for the send operation. Please note that INPUT_DATA should be listed under “Input signals” instead of “Output signals”

The receiving process shall operate as specified by the following timing diagram:

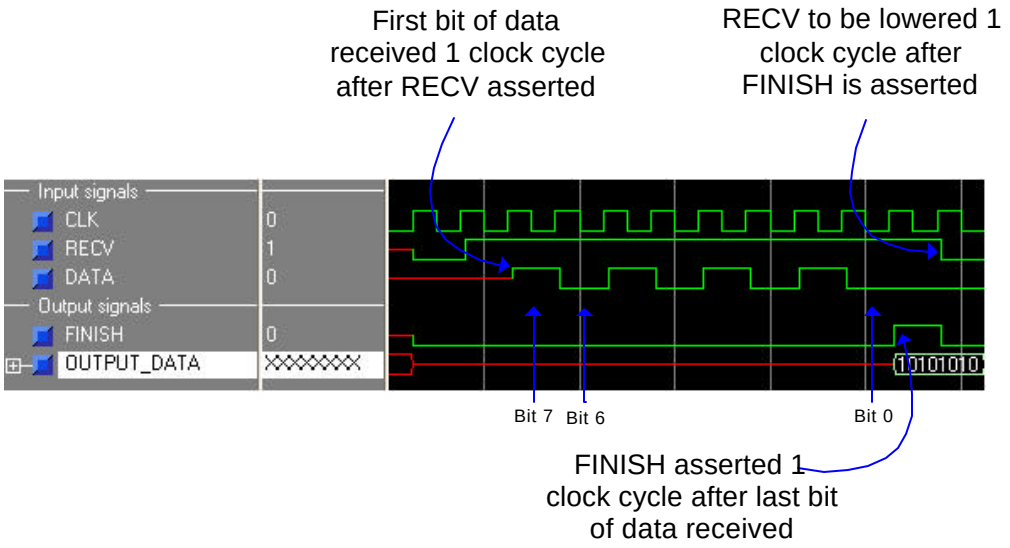

Figure 4: Timing diagram for the receive operation.

Please note that when specifications are given for input signals, they give the timing of the signals as they will appear in the testbench. When specifications are given for output

signals, these timings must be enforced by the system that you design. Also, note that DATA acts as an output for the send operation and an input for the receive operation.

The specific requirements are as follows:

1) You are required to implement and test your design in a modular manner. For instance, if you decide to use a shift register, you should create a shift register entity and architecture, as well as a test bench. Each major functional unit should be implemented as a module.
2) For the top-level system, you will be provided with test benches in order to test your system. This is why it is very important you define your input and output signals as described above.
3) All simulations conducted as part of this project should be functional simulations.
4) Designs will be grades based on creativity of solution, complexity of implementation (simpler is better), and explanation of how the design works. You should aim to incorporate what you have learned from other projects into different parts of the design when appropriate.

When you turn in your project report, it should include the following:

1) A description of your design. This includes block diagrams and descriptions of each module within your design, as well as how requirements of the design (as presented in this project description) are fulfilled.
2) A description of the test plan. A test plan and test bench should be presented for each module. For top-level system testing, please include a description of what the provided test benches test. If you feel that your top-level system requires further testing, please feel free to develop additional test plans and test benches to test these other functions.
3) Timing diagrams for each testing simulation performed. Please include any data necessary to interpret the waveform, as well as a description of what the expected results are for each test case within the simulation as well as how your results compare to what is expected.
4) A copy of all of the VHDL code used in the implementation and testing of this project.
5) Answers to the following questions:
    1. A tri-state buffer is used on each end of the data link to control the direction of data flow between two different units. However, if both tri-state buffers are set to their high-impedance state, each module will read in an undefined level as the input half of the bidirectional port. If we were going to physically build a circuit such as this, what could we add to the circuit to prevent this undefined state from occurring in this situation? (Hint: a similar situation occurs with open-collector or open-drain devices)
    2. Most microprocessors provide functionality similar to the shift register (in both shift directions) as a basic instruction at the machine language level. In terms of arithmetic operations, why do you think this functionality is provided? In other words, what arithmetic function is performed through a shift process?