EECE 359 – Computer Communications and Networking **Spring 2020**

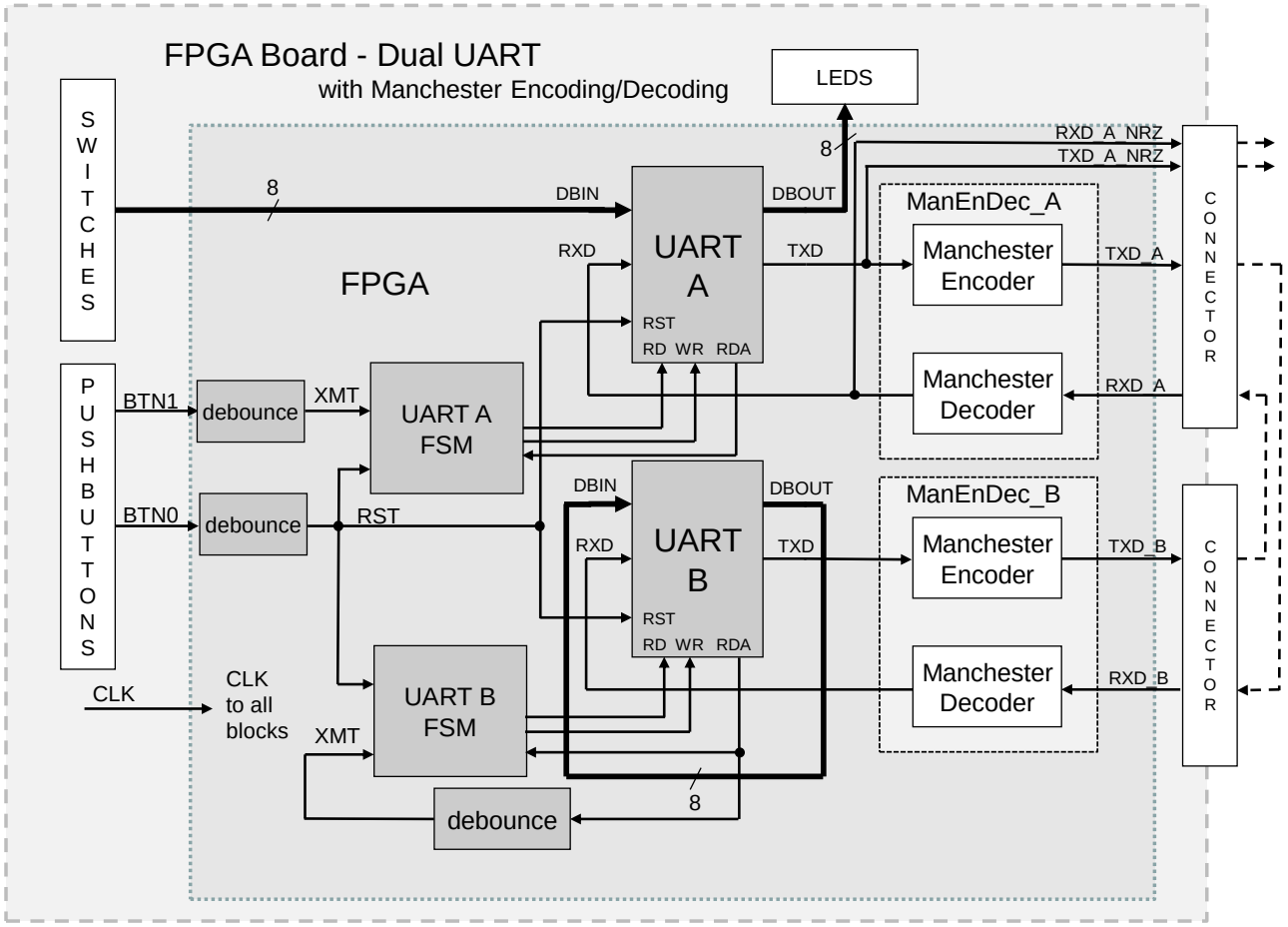
Lab #3 – Data Link using Manchester Encoding/Decoding (2 Week Lab)

Objectives

This lab builds on the previous lab. In this lab you will develop a Manchester encoder and decoder. The encoder and decoder will be added between each UART and its external connections as shown in the block diagram below. You will design a VHDL module for the encoder and one for the decoder, plus a higher level encoder/decoder module. You will synthesize the hardware design, verify your design with simulation, integrate them with the UARTs, simulate again, and finally implement the new design on the FPGA board and demonstrate its operation.

System Description

The system in the previous lab communicated using binary NRZ signals. In the modified system for this lab, the Manchester Encoder will convert the binary NRZ signals to Manchester biphase coded signals. On the receiving end, a Manchester decoder will convert the biphase code back to binary NRZ.



Design Hierarchy/Files

Dual UART design with Manchester Encoder/Decoder

Implementation

- dualUART – Behavioral (dualUART .vhd)
  - UART_A – UART – Behavioral (UART.vhd)
    - Inst_UART_Tx – UART_Tx – Behavioral (UART_Tx.vhd)
    - Inst_UART_Rx – UART_Rx – Behavioral (UART_Rx.vhd)
  - UART_B – UART – Behavioral (UART.vhd)
    - Inst_UART_Tx – UART_Tx – Behavioral (UART_Tx.vhd)
    - Inst_UART_Rx – UART_Rx – Behavioral (UART_Rx.vhd)
  - UART_A_FSM – UART_FSM – Behavioral (UART_FSM.vhd)
  - UART_B_FSM – UART_FSM – Behavioral (UART_FSM.vhd)
  - ManEnDec_A – Manchencded – med_beh (med.vhd)
    - ManEncoder – Manchencoder – me_beh (me.vhd)
    - ManDecoder – Manchdecoder – md_beh (md.vhd)
  - ManEnDec_B – Manchencded – med_beh (med.vhd)
    - ManEncoder – Manchencoder – me_beh (me.vhd)
    - ManDecoder – Manchdecoder – md_beh (md.vhd)
  - RST_debounce – debounce – Behavioral (real_debounce.vhd)
  - XMT_A_debounce – debounce – Behavioral (real_debounce.vhd)
  - XMT_B_debounce – debounce – Behavioral (real_debounce.vhd)
  - dualUART-PapilioDuo.ucf

Manchester Encoder/Decoder Simulation Environment (for Part B)

- Test_med – Behavioral (Test_med .vhd) – Test bench
  - uut – Manchencdec – med_beh (med.vhd)
    - ManEncoder – Manchencoder – me_beh (me.vhd)
    - ManDecoder – Manchdecoder – md_beh (md.vhd)

Manchester Encoder/Decoder Simulation Environment (for Part C)

- Test_dualUART – behavior (Test_dualUART.vhd)
  - uut - dualUART – Behavioral (dualUART .vhd)

All of the code for the FPGA in this lab is re-used from Lab 2, except **med.vhd**, **me.vhd** and **md.vhd**. The file med.vhd is provided for you and me.vhd and md.vhd contain templates to guide you through the development of the Manchester Encoder and Manchester Decoder, respectively. The med.vhd file combines your Manchester encoder and decoder modules into a single higher level module and divides the system clock appropriately for the desired Baud rate. You will modify your **dualUART.vhd** file to integrate two Manchester encoder/decoder modules into the system. The **Test_med.vhd** test bench file is also

provided for you. This file is used to test your encoder and decoder designs with the simulator.
This lab has three parts:

Part A: Develop the Manchester Encoder and test it using simulation
Part B: Develop the Manchester Decoder and test it with the Encoder using simulation
Part C: Integrate the Manchester Encoder and Decoder with the UART system from Lab 2, verify with simulation, and then demonstrate the system on the FPGA board

PART A

The first part of this lab consists of the development and simulation of the Manchester encoder.

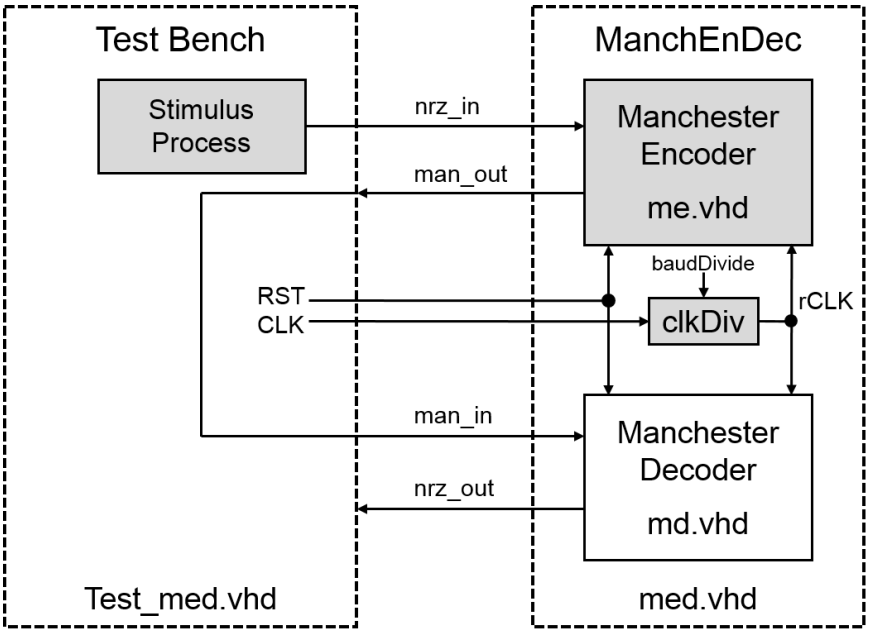
Create a Xilinx ISE Project and complete the design of the Manchester Encoder

- ☐ Use the "**File => Copy Project…**" option in ISE Project Navigator to save your Lab2 project as a new project called Lab3. *Be sure that the working directory is on the C drive*.
- ☐ Add the source file **me.vhd** to the project.
- ☐ The comments provided in the me.vhd template will guide you through the development of the VHDL code for the Manchester encoder.
- ☐ Your code will include an **Encoder FSM**. Draw a state diagram showing all states, transitions, and outputs. Submit your FSM state diagram as a file to the lab assignment (preferred) or give a hard copy to the TA

Simulate your Manchester Encoder using iSim

- ☐ Add the source file **med.vhd** and the test bench file **Test_med.vhd** to your Lab3 project.
- ☐ In the test bench, set the value of the **clk_period** constant to the period of the system clock on your FPGA board.
- ☐ Run a simulation of the Manchester encoder/decoder module using the Test_med.vhd test bench. See the diagram below. Disregard the warning messages due to the missing decoder module.
- ☐ Group the test bench signals in the waveform viewer using the name "**Test Bench**".
- ☐ In addition to the test bench signals, add all of the signals for the ManEncoder module to the waveform viewer, group them with the name "**ManEncoder**".
- ☐ Change the radix of the **num_bits_sampled** signal to "unsigned decimal".
- ☐ Save your waveform configuration to a waveform configuration file, **Test_med.wcfg**.
- ☐ Re-run the simulation for 65 microseconds.

- ☐ You should see that the test bench generated four frames on the ***nrz_in*** input to the Manchester encoder. The signal on the encoder output, ***manout***, should be the input NRZ data converted to the Manchester code for that data.
- ☐ Determine by inspection if the ***manout*** signal is correctly encoding the data of the ***nrz_in*** signal to Manchester <u>for all four frames</u>. If not, debug and make the necessary corrections.



Demonstrate the Manchester Encoder operation using the simulator

- ☐ Explain the operation of your Manchester Encoder to a lab proctor and demonstrate it using the simulator.

  <u>Get checked off by a lab proctor on their sheet</u>
- ☐ <u>Submit, to the lab assignment, a PDF of your simulation waveforms for the Manchester Encoder, showing all of its signals and with the ***sampling_ctr*** signal expanded to show every bit</u>.
- ☐ <u>Submit your Manchester Encoder VHDL code (me.vhd) to the lab assignment.</u>

Lab Submission Summary for PART A

- ☐ Manchester Encoder FSM state diagram
- ☐ Encoder simulation checkoff on proctor's sheet
- ☐ PDF of encoder simulation waveforms
- ☐ Manchester Encoder VHDL code (me.vhd)

PART B

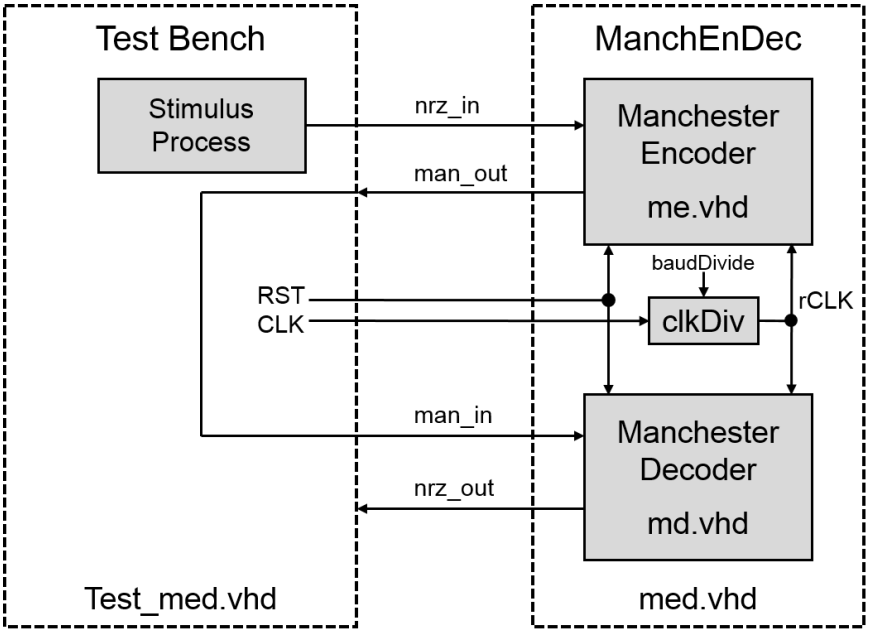
In this part of the lab you will develop the Manchester decoder, integrate the Manchester decoder with the Manchester encoder, and verify their operation together through simulation. The provide test bench (Test_med.vhd) will be used to demonstrate the full operation of the Manchester Encoder and Decoder.

Complete the design of the Manchester Decoder

- ☐ Add the **md.vhd** source file to your Lab3 project.
- ☐ The comments provided in the md.vhd template will guide you through the development of the VHDL code for the Manchester decoder.
- ☐ The decoder code includes a Start Detector FSM for detecting the start bit and producing the data_frame signal. Draw the state diagram for the start bit detector FSM. Submit your FSM as a file to the lab assignment (preferred) or give a hard copy to the TA.

Simulate your Manchester Decoder

The **Test_med.vhd** test bench along with the **med.vhd**, **me.vhd**, and **md.vhd** files will create the test environment shown in the figure below. In this simulation environment, the system consists of your designs for the Manchester encoder and decoder placed in a



higher level module (med.vhd). The test bench, Test_med.vhd, used to generate input data, completes the system as illustrated.

- ☐ In the med.vhd file, set the **baudDivide** constant to "**000000000**".
- ☐ Run the simulation and open your Test_med.wcfg configuration file.

- ☐ In addition to signals you had in part A, add all of the signals for the ManDecoder module to the waveform viewer, and group them with the name "**ManDecoder**".
- ☐ Set the radix of the ***bit_counter*** signal to "unsigned decimal".
- ☐ Expand the ***clkdiv*** signal to show the individual bits of the counter.
- ☐ Save your waveform configuration to the **Test_med.wcfg** file.
- ☐ Re-launch the test bench, running it for 65 microseconds.
- ☐ Determine by inspection if the ***nrz_out*** signal is being correctly decoded from the ***manin*** signal. That is, verify that the decoder is correctly converting the Manchester coded signal to NRZ. The easiest way to verify this is to compare the ***nrz_out*** signal with the ***nrz_in*** signal. They should look the same, except for a time delay difference. If the signals don't match, debug and make the necessary corrections.

Demonstrate the Manchester Decoder operation using the simulator with the provided test bench

- ☐ Run the simulation for 65 microseconds using the Test_med.wcfg waveform configuration file.
- ☐ Explain the operation of your Manchester Decoder to a lab proctor and demonstrate it using the simulator.

  Get checked off by a lab proctor on their sheet
- ☐ Submit, to the lab assignment, a PDF of your simulation waveforms, with the decoder waveform group expanded and the test bench and encoder waveform groups collapsed. Be sure the ***clkdiv*** signal is also expanded.
- ☐ Submit your Manchester Decoder VHDL code (md.vhd) to the lab assignment.

Lab Submission Summary for PART B

- ☐ Start Detector FSM state diagram
- ☐ Decoder simulation checkoff on proctor's sheet
- ☐ PDF of decoder simulation waveforms
- ☐ Manchester Decoder VHDL code (md.vhd)

PART C

In this part of the lab you will integrate the Manchester encoder/decoder with the dualUART system, verify with simulation, and demonstrate it on the FPGA board.

Integrate the Manchester Decoder with the dualUART system

- ☐ Instantiate two Manchester encoder/decoder modules in the dualUART module. This can be done using the following process:
    a. In the View pane of the *Design* panel, select *Implementation*.
    b. In the *Hierarchy* pane, select the source file for which you want to create an instantiation template. In this case it is the module *manchencdec*.
    c. In the *Processes* pane, expand *Design Utilities*, and double-click the *View HDL Instantiation Template* process.
    d. This will open a read only file in the editor pane containing the code for the component declaration of the module and a code template for instantiating the module. You can copy and paste these to the appropriate places in the **dualUART** module.
    e. Label the two instantiations with the names **ManEnDec_A** and **ManEnDec_B**.
- ☐ Now, connect the two modules up with the UARTs and the I/O ports, including the two new ports just added to the **dualUART** top-level. Declare new signals as needed to connect each of the two new ***ManchEncDec*** modules into the system.
- ☐ Add two output ports (**RXD_A_NRZ** and **TXD_A_NRZ**) to the dualUART entity as shown in the system diagram and connect them so that the signals on UART_A's RXD and TXD ports can be viewed on the oscilloscope using the new dualUART output ports.
- ☐ Check for syntax errors and then run synthesis. The warnings listed below are expected. If there are no errors or suspicious warnings, move on to simulation.

	Synthesis Messages - Errors, Warnings, and Infos
WARNING	Xst:2677 - Node <UART_A/Inst_UART_Rx/OEint> of sequential type is unconnected in block <dualUART>.
WARNING	Xst:2677 - Node <UART_A/Inst_UART_Rx/PEint> of sequential type is unconnected in block <dualUART>.
WARNING	Xst:2677 - Node <UART_B/Inst_UART_Rx/OEint> of sequential type is unconnected in block <dualUART>.
WARNING	Xst:2677 - Node <UART_B/Inst_UART_Rx/PEint> of sequential type is unconnected in block <dualUART>.
WARNING	Xst:1293 - FF/Latch <UART_B/clkDiv_7> has a constant value of 0 in block <dualUART>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <UART_B/clkDiv_8> has a constant value of 0 in block <dualUART>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <UART_A/clkDiv_7> has a constant

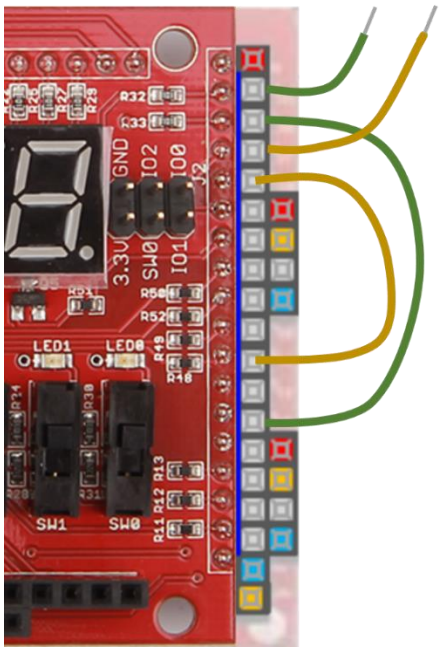
	value of 0 in block <dualUART>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <UART_A/clkDiv_8> has a constant value of 0 in block <dualUART>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <ManEnDec_B/clkDiv_7> has a constant value of 0 in block <dualUART>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <ManEnDec_B/clkDiv_8> has a constant value of 0 in block <dualUART>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <ManEnDec_A/clkDiv_7> has a constant value of 0 in block <dualUART>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <ManEnDec_A/clkDiv_8> has a constant value of 0 in block <dualUART>. This FF/Latch will be trimmed during the optimization process

Simulate the Manchester Encoders and Decoders with the two UARTs

- ☐ You will have to add the two new output ports of the dualUART to its component declaration and instantiation in the **Test_dualUART.vhd** file.
- ☐ Set the **baudDivide** constants in the **UART.vhd** and **med.vhd** files to the value required for a *9600 Baud data rate* on your FPGA board.
- ☐ Run iSim using the Test_dualUART test bench for 3.5 milliseconds.
- ☐ Add the waveforms for the **ManEnDec_A** and **ManEnDec_B** modules and re-launch the simulation.
- ☐ Verify that the test bench waveforms for signals ***txd_a_nrz*** and ***rxd_a_nrz*** both have frames with the data pattern “00110101011”, but separated in time.
- ☐ Submit a PDF of your waveforms for this final simulation to the lab assignment. The view should show the **Test Bench** waveforms and the **two Manchester encoder/decoder top-level module** waveforms for 3.5 milliseconds.

Test out the complete system on your FPGA board

- ☐ Add the two new output ports added to the dualUART entity to your **UCF** file. The pinouts for the Papilio Duo board are shown below.



5V		
D0	TXD_A_NRZ	
D1	TXD_A	
D2	RXD_A_NRZ	
D3	RXD_A	
D4		5V
D5		3.3V
D6		
D7		GND
D8		
D9	TXD_B	
D10		
D11	RXD_B	
D12		5V
D13		3.3V
D14		
D15		GND
GND		
3.3V		

- ☐ Implement the design and generate a programming file
- ☐ Load your FPGA board using the programming file
- ☐ Set up the board connections as you did for Lab 2. Verify that the system operates properly. It should operate the same as it did in Lab 2 as viewed from the switches and lights. However, the signals when viewed on the scope will be Manchester coded.
- ☐ View both the Manchester coded signals and the NRZ signals provided by the dualUART on the oscilloscope using all four channels. You will need to synchronize (trigger) the scope with an NRZ signal since the Manchester signals are always transitioning.
- ☐ Demonstrate your FPGA board to a lab proctor using several patterns of data on the switches. Also, show the Manchester waveforms on the oscilloscope.

  Get checked off by a lab proctor on their sheet

- ☐ Save an image of the scope view to a file and submit it to the lab assignment.
- ☐ Submit your dualUART code (dualUART.vhd) to the lab assignment

Lab Submission Summary for PART C

- ☐ PDF of final simulation waveforms for the dual UART system with Manchester signal coding
- ☐ Checkoff for demonstration of FPGA board by a lab proctor on their sheet
- ☐ Image of the four signals (TXD_A, TXD_B, TXD_A_NRZ, TXD_B_NRZ) on the scope
- ☐ dualUART VHDL code (dualUART.vhd)