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

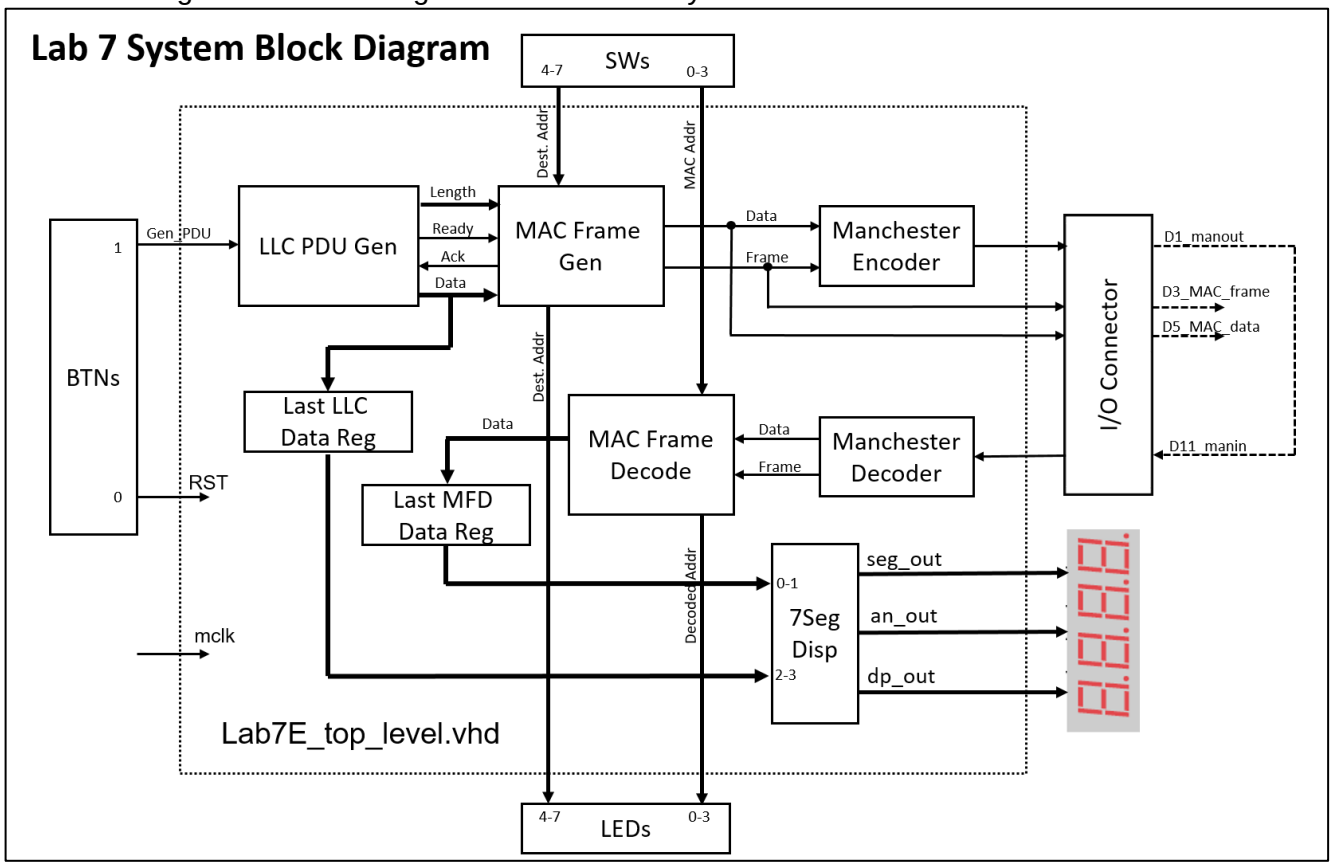
Lab #7 – Ethernet MAC Protocol

Objectives

The objective of this lab is to develop a system that implements a simplified version of the Ethernet MAC protocol in hardware and demonstrates its operation. The system includes a module that generates simulated client data from the Logical Link Control layer of protocol as well as Manchester encoder and decoder modules to provide the physical layer beneath the MAC layer.

System Description

The block diagram below is a high-level view of the system.



In this system, implemented on the FPGA Board, pushbutton 0 will reset the board. Pushbutton 1 will cause the *LLC PDU Generator* to generate a single LLC PDU of random length between 1 and 32 octets (bytes). The LLC PDU will contain random data. Switches 4-7 provide the 4 LSBs of the destination address to be inserted into the MAC frame. Switches 0-3 provide the 4 LSBs of the MAC address for the receiving end. LEDs 4-7 display the 4 LSBs of the destination address and LEDs 0-3 display the 4 LSBs of the MAC address. There are two 8-bit data registers. One stores the last byte of the LLC PDU generated and the other stores the last byte of the LLC PDU received by the *MAC Frame Decoder*. These registers are displayed in hexadecimal on the left two characters and the right two characters, respectively, on the 7-segment displays.

The lab is divided into five parts. Each of the parts has its own simulation test bench. In the 5th part, the system is implemented on the FPGA Board.

This lab carries a grading weight of 3 labs.

PART A – *LLC PDU Generator* Design

Design Hierarchy/Files

Test_Lab7A – behavior (Test_Lab7A.vhd)

uut – Lab7A_top_level – Behavioral (Lab7A_top_level.vhd)

RST_debounce – debounce – Behavioral (real_debounce.vhd)

Gen_PDU_debounce – debounce – Behavioral (real_debounce.vhd)

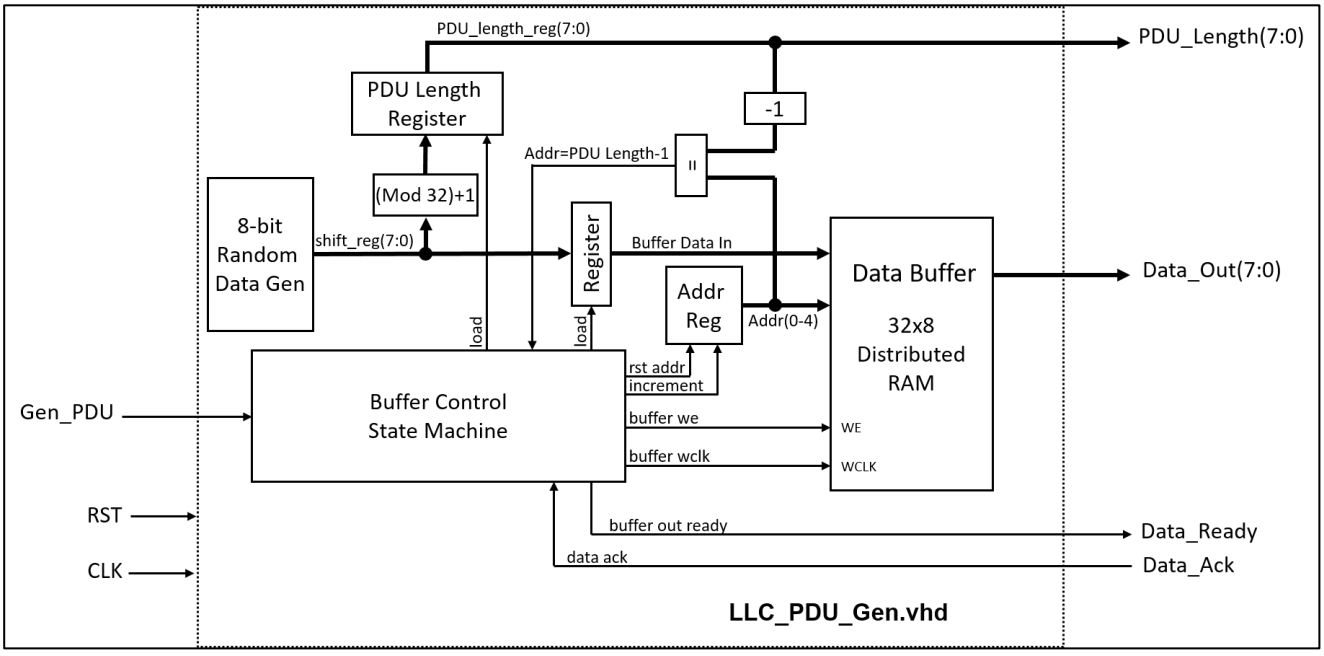
LLC_PDU_GenA – LLC_PDU_Gen – Behavioral (LLC_PDU_Gen.vhd)

LLC_Data_Buffer – Data_Buffer – Behavioral (Data_Buffer .vhd)

For this part of the lab, you will develop the *Data Buffer* (Data_Buffer.vhd) and the *LLC PDU Generator* (LLC_PDU_Gen.vhd). Templates are provided for those files. You can reuse your code from RandBitGen.vhd in Lab 5 with modifications for the random data generator required in the *LLC PDU Generator*. Note that you will not directly copy the RandBitGen.vhd from Lab 5 to the Lab7 project. You will simply copy and paste code for the random bit generator shift register from RandBitGen.vhd to the LLC_PDU_Gen.vhd file. The remaining files including the test bench are provided.

LLC PDU Generator Module Description

A block diagram for this module is shown below. This module will generate simulated PDUs from the Logical Link Control layer of protocol. The contents of the PDU will be random data generated by the



Random Data Generator. The length of each PDU will be defined by the contents of the PDU Length Register and will have a range of 1 to 32 octets.

The random PDU data will be written into the *Data Buffer* which will be implemented using Distributed RAM in the FPGA. When an LLC PDU is ready for the *MAC Frame Generator* (the next module

following this one in the system) to take the data, the data will be read out of the Distributed RAM as needed by the *MAC Frame Generator*.

You will develop a Buffer Control FSM to control the writing of data to the buffer as well as the reading of data from the buffer. A Data Ready/Data Acknowledge "handshake" scheme will be used to control the transfer of data from the *Data Buffer* in the *LLC PDU Generator* to the *MAC Frame Generator* modules.

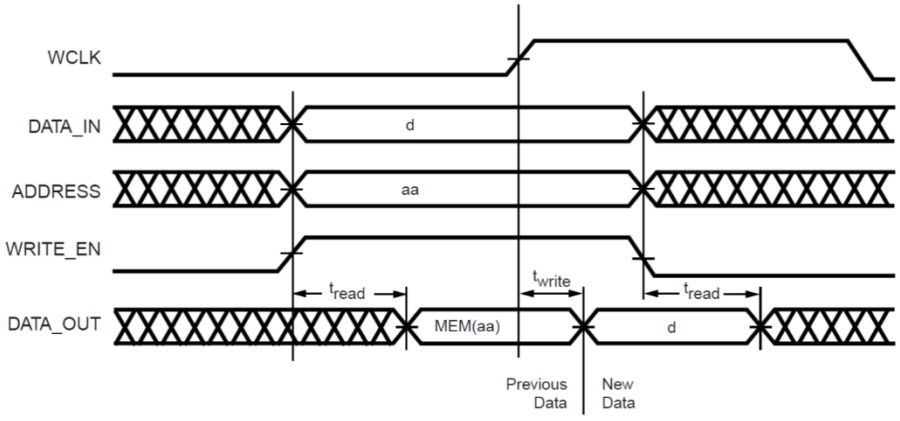
Creating the *Data Buffer* using Distributed RAM in the FPGA

There are two ways write the VHDL code for a distributed RAM. One way is to use device primitive instantiations. That is, using an appropriate technology specific instantiation template from the language template library, instantiate the required number of template copies for the desired size of the RAM, and then connecting them up. Though fairly straight forward, this method generally results in a large amount of code.

The second way to create distributed RAM in VHDL is to implement the RAM by inference. Inference code is code that follows a specific structure which is recognized by most synthesis tools and thus is technology independent. This type of code was used to implement the register file that was used in DSD Lab5. A code template suitable for this lab is provided with this procedure on the course website.

Data Buffer Timing

The required timing for writing and reading distributed RAM is shown below. The WRITE_EN is required to be active when writing to the RAM. When reading the RAM the WRITE_EN should be kept low. The ADDRESS and DATA_IN need to be stable when the rise edge of WCLK occurs for writing to the RAM. The ADDRES and DATA_IN also need to be stable before the DATA_OUT can be used. You will design the Buffer Control FSM to provide the necessary timing.



Design your Buffer Control State Machine to control the writing of a block of data to the *Data Buffer* and then control the reading of the block of data from the *Data Buffer* using a Data Ready/Data Acknowledge handshake.

Reset should put the state machine into the Idle State. The Gen_PDU input should begin the sequence of states to create the timing for writing PDU-length bytes (octets) of data from the Random Bit Generator to the *Data Buffer*.

When the data is finished being written to the *Data Buffer*, the state machine shall cause the Data Ready to become active. The state machine shall then control the reading of the *Data Buffer* to provide the data to the *MAC Frame Generator* using the Data Ready/Data Acknowledge handshake. The *MAC Frame Generator* activates the Acknowledge after it has loaded the data from the *Data Buffer* into its shift register. It will hold the Acknowledge signal high until the Data Ready signal goes low.

Part A Procedure:

- **Create a project named "Lab7" in the ISE Project Navigator**
- **Add the provided Lab7 Part A VHDL files to your project**
- **Complete the design of the *Data Buffer* and *LLC PDU Generator***
- **Draw a state diagram showing each state of the FSM, transition conditions, and outputs (submit as a file in the lab assignment or a hardcopy to the TA)**
- **Run "Synthesize - XST" on the top level module (Lab7A_top_level) to check for Errors and Warnings**

  There should be no errors and only two warnings when running synthesis. The following messages are expected:

	Synthesis Messages - Errors, Warnings, and Infos
INFO	Xst:3218 - HDL ADVISOR - The RAM <Mram_RegFile> will be implemented on LUTs either because you have described an asynchronous read or because of currently unsupported block RAM features. If you have described an asynchronous read, making it synchronous would allow you to take advantage of available block RAM resources, for optimized device usage and improved timings. Please refer to your documentation for coding guidelines.
WARNING	Xst:1293 - FF/Latch <PDU_length_reg_6> has a constant value of 0 in block <LLC_PDU_Gen>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <PDU_length_reg_7> has a constant value of 0 in block <LLC_PDU_Gen>. This FF/Latch will be trimmed during the optimization process.

  Because we limited the range of the PDU length values, bits 6 and 7 of that register are always zero. If you have any other error or warning messages, make the appropriate corrections to your VHDL code.

- **Use simulation to verify correct operation of the *LLC PDU Generator* with the *Data Buffer***

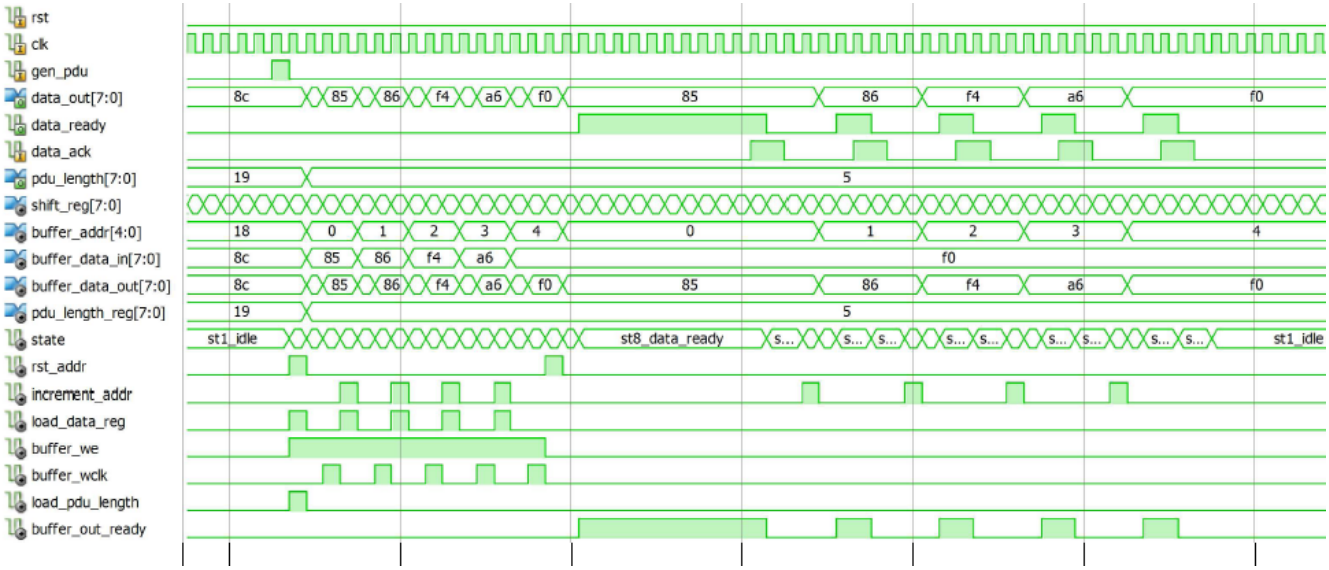
  1. Use the provided test bench file Test_Lab7A.vhd.
  2. Run the test bench for ~30 microseconds. Adjust this time so that you can see FOUR full transfers of PDU data.
  3. Group the test bench signals.
  4. Add all of the signals in the *LLC_PDU_Gen* module to the waveform viewer and assign them to a new group.
  5. Save the waveform configuration to "test_lab7a.wcfg".

  The test bench file, Test_Lab7A.vhd, contains stimulus code to continuously generate LLC PDU data frames. The test bench provides the handshake signal *data_ack* in response to *data_ready*. When you demonstrate the operation of your *LLC PDU Generator* to the TA, show that four

complete data transfers occur and that the number of data bytes transferred matches the LLC_PDU_length in each of the four frames.

An example of the waveforms for this test bench is shown below. Because the timing of your state machine may vary from the example, and since the *LLC_PDU_length* is a random number, your results will likely be different.

Example simulation waveforms:



- **<u>Demonstrate the operation of your *LLC PDU Generator* to a lab proctor and get checked off</u>**

  Evaluation Criteria for Simulation Results:

  1. Number of Buffer RAM write cycles is equal to the value in the PDU_length_reg.
  2. Number of Buffer RAM read cycles is equal to the value in the PDU_length_reg.
  3. The data values read from the Buffer RAM correspond to the values written to the Buffer RAM (i.e. the first value written equals the first value read, etc.). Changing the radix (right click on waveform) to hexadecimal makes the comparisons of values easier to see.
  4. Data_ready and data_ack timing looks “clean”. For example, the data_ready should go low when the data_ack goes high.
  5. Verify 1-4 for all four frames generated by the test bench.

- **<u>Submit TWO PDFs of your simulation waveforms to the lab assignment</u>**

  Both images are to show all the signals in the *LLC_PDU_Gen* module group.

  1. The <u>first image</u> is to show FOUR generated PDUs
  2. The <u>second image</u> is to be zoomed into ONE PDU generation cycle. Choose one with a shorter length (say between 2 and 8). Include the time period from the leading edge of the PDU_Gen pulse to the last data ready/acknowledge cycle before the next PDU_Gen pulse.

  Note: the easiest way to zoom is to place the first cursor line on the leading edge of the PDU_Gen pulse and the second cursor line (shift – left click) past the falling edge of the last data_ack, and then click on the red magnifier icon (zoom to cursors).

- **<u>Submit the following code files</u>:** LLC_PDU_Gen.vhd and Data_Buffer .vhd

PART B – *MAC Frame Generator* Design

Design Hierarchy/Files

Test_Lab7B – behavior (Test_Lab7B.vhd)

uut – Lab7B_top_level – Behavioral (Lab6B_top_level.vhd)

RST_debounce – debounce – Behavioral (real_debounce.vhd)

Gen_PDU_debounce – debounce – Behavioral (real_debounce.vhd)

LLC_PDU_GenA – LLC_PDU_Gen – Behavioral (LLC_PDU_Gen.vhd)

LLC_Data_Buffer – Data_Buffer – Behavioral (Data_Buffer .vhd)

MAC_Frame_GenA – MAC_Frame_Gen – Behavioral (MAC_Frame_Gen.vhd)

FCS_Gen – CRC – Behavioral (CRC.vhd)

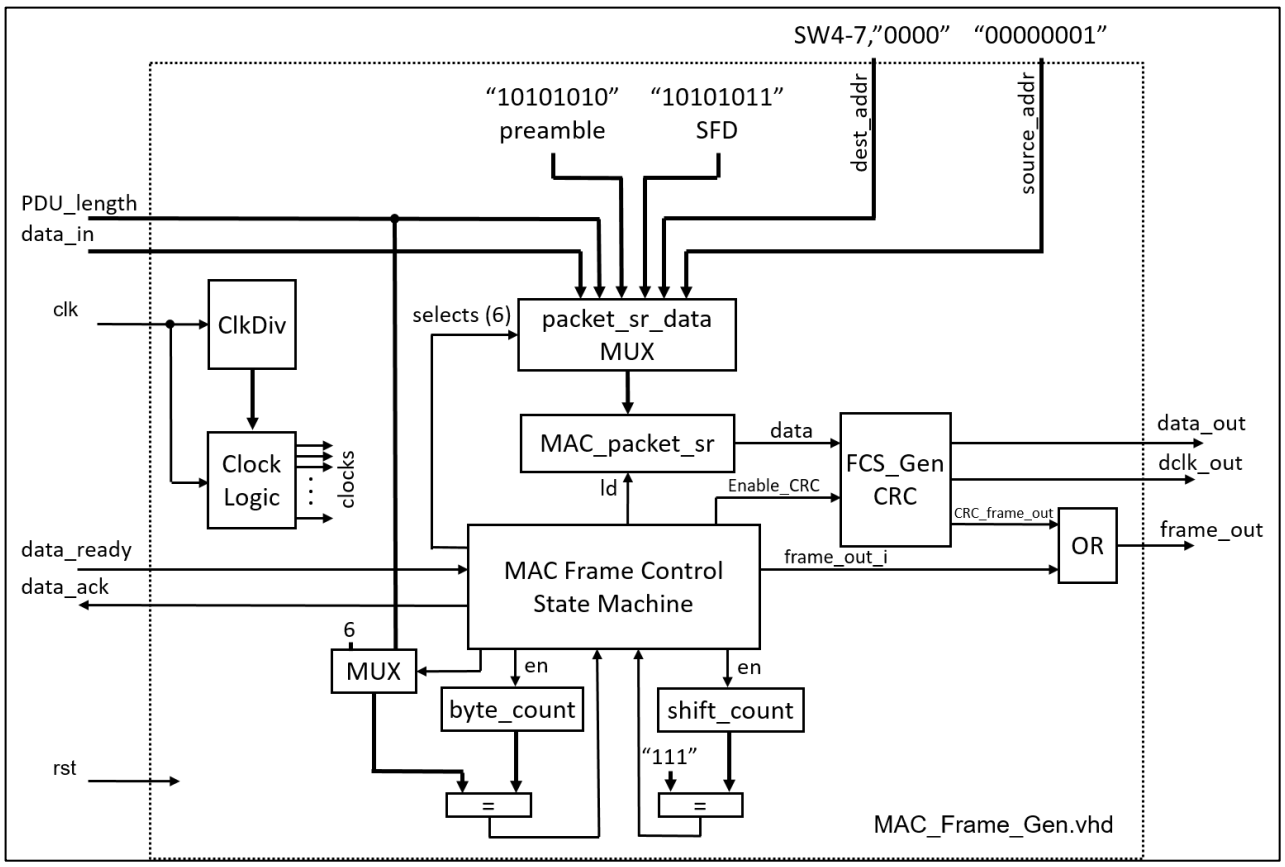
For this part of the lab, you will be developing the *MAC Frame Generator* (MAC_Frame_Gen). A template is provided for the file. You will reuse your CRC.vhd file developed in Lab 4 for the FCS generator used in the *MAC Frame Generator*. The remaining files are provided or were provided in the previous part of the lab.

Module Description

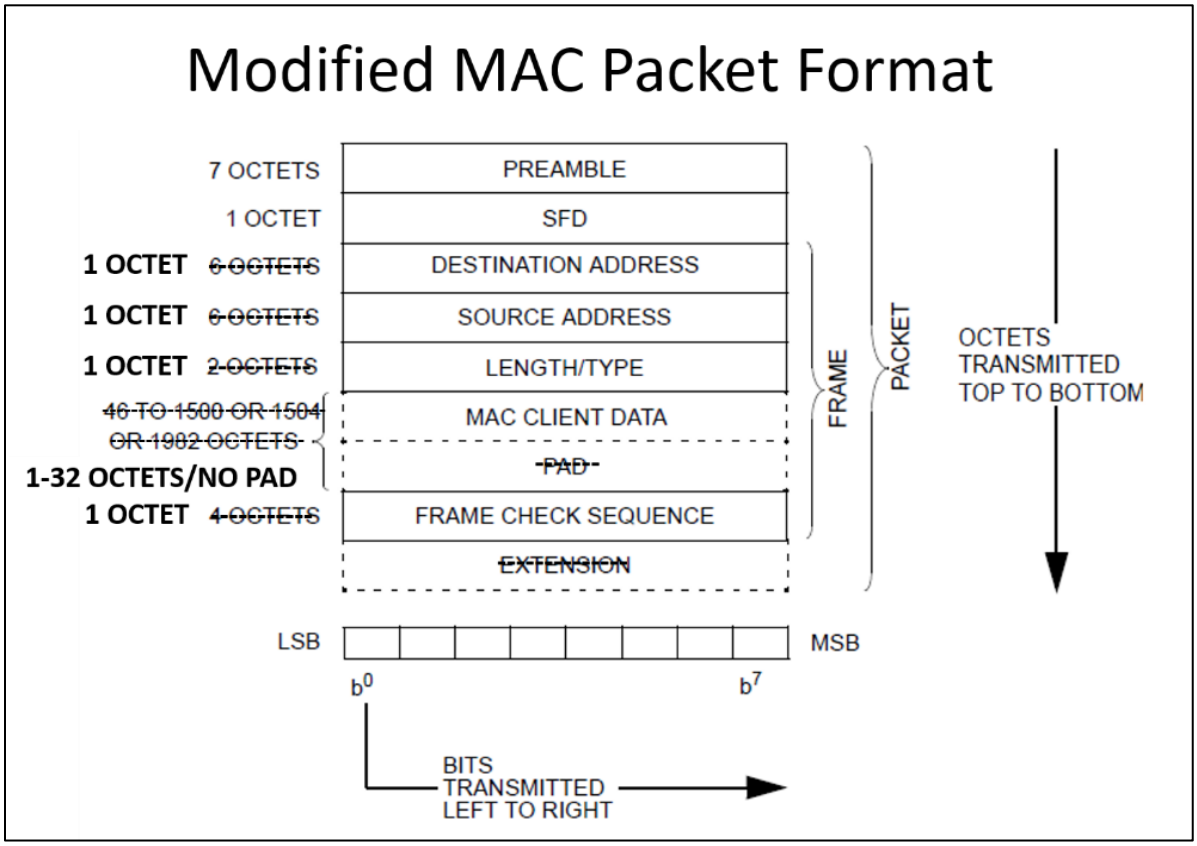
The function of the *MAC Frame Generator* is depicted in the block diagram below (on next page).

This module receives the PDU Length and the PDU data for the LLC PDU from the *LLC PDU Generator* Module. Using that data as the client data for the MAC frame, the *MAC Frame Generator* assembles the frame and outputs the frame serially. The size of some of the fields of the 802.3 MAC packet format are reduced for simplification as follows:

- Preamble – 7 octets (no change)
  10101010 10101010 10101010 10101010 10101010 10101010 10101010
- SFD (Start Frame Delimiter) – 1 octet (no change)
  10101011
- Destination Address – 1 octet
- Source Address – 1 octet
- Length/Type – 1 octet (length = 1 to 32)
- MAC Client Data – 1 to 32 octets of random data
- No pad
- Frame Check Sequence – 1 octet
- No Extension field



The modifications are shown in the figure below.



The destination address least significant bits are from switches 4-7, and the higher order bits are set to zero. The source address is hardcoded as 00000001. The Data Select, controlled by the state machine, selects the data sources to be loaded into the the MAC_packet_sr (shift register), 8 bits at a time to be shifted out as the MAC frame. The *FCS_Gen* uses the CRC module to append an FCS to the end of the data frame.

The output bit rate is to be arbitrarily set using a data bit period of 640 ns.

Part B Procedure:

- **Continue using your project named "Lab7" in the ISE Project Navigator**
- **Add the provided Lab7 Part B VHDL files to your project**
- **Complete the design of the *MAC Frame Generator*.**
- **Draw a state diagram showing each state of the FSM, the inputs used for transition decisions, and the outputs produced**
  (Submit as a file in the lab assignment or a hardcopy to the TA)
- **Run "Synthesize - XST" on the top level module (Lab7B_top_level) to check for Errors and Warnings**

  There should be no errors and only five warnings when running synthesis. The following messages are expected:

	Synthesis Messages - Errors, Warnings, and Infos
INFO	Xst:3210 - "C:\Users\Carl\Documents\EECE 359 - Computer Networks\LABS - PAPILIO DUO\Lab7 - Ethernet MAC Protocol\Source Code\MAC_Frame_Gen.vhd" line 137: Output port <error_out> of the instance <FCS_Gen> is unconnected or connected to loadless signal.
WARNING	Xst:647 - Input <P<0:0>> is never used. This port will be preserved and left unconnected if it belongs to a top-level block or it belongs to a sub-block and the hierarchy of this sub-block is preserved.
WARNING	Xst:647 - Input <P<8:8>> is never used. This port will be preserved and left unconnected if it belongs to a top-level block or it belongs to a sub-block and the hierarchy of this sub-block is preserved.
INFO	Xst:1799 - State det is never reached in FSM <state>.
INFO	Xst:1799 - State det_calc is never reached in FSM <state>.
INFO	Xst:1799 - State check_remainder is never reached in FSM <state>.
INFO	Xst:1799 - State report_error is never reached in FSM <state>.
INFO	Xst:3218 - HDL ADVISOR - The RAM <Mram_RegFile> will be implemented on LUTs either because you have described an asynchronous read or because of currently unsupported block RAM features. If you have described an asynchronous read, making it synchronous would allow you to take advantage of available block RAM resources, for optimized device usage and improved timings. Please refer to your documentation for coding guidelines.
WARNING	Xst:1710 - FF/Latch <error_out> (without init value) has a constant value of 0 in block <CRC>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1293 - FF/Latch <PDU_length_reg_6> has a constant value of 0 in block <LLC_PDU_Gen>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <PDU_length_reg_7> has a constant value of 0 in block <LLC_PDU_Gen>. This FF/Latch will be trimmed during the optimization process.

INFO	Xst:2261 - The FF/Latch <MAC_Frame_GenA/CRC_data_in_d1> in Unit <Lab7B_top_level> is equivalent to the following FF/Latch, which will be removed : <MAC_Frame_GenA/FCS_Gen/data_in_d1>
INFO	Xst:2261 - The FF/Latch <MAC_Frame_GenA/CRC_data_in_d2> in Unit <Lab7B_top_level> is equivalent to the following FF/Latch, which will be removed : <MAC_Frame_GenA/FCS_Gen/data_in_d2>

The warnings include the two from Part A because bits 6 and 7 of the PDU_length_reg will always be zero because we limit the value to be in the range of 1 to 32. You should also get three warning messages due to the CRC module. Two are because the first and last bits of P are not used. The third one is because the CRC is only used for FCS generation and therefore the error_out signal is not used. If you have other error or warning messages, make the appropriate corrections to your VHDL code.

- **Use simulation to verify correct operation of the *MAC Frame Generator* with the FCS Generator**

    1. Use the provided test bench file Test_Lab7B.vhd.
    2. Run the test bench for around 300 to 750 microseconds. Use a simulation time that produces two full cycles of the "frame_out" signal (amount of time will depend upon amount of data in the two frames).

- **Demonstrate the operation of your *MAC Frame Generator* to a lab proctor and get checked off**

Evaluation Criteria for Simulation Results:

1. Should see two MAC packets (frames)
2. *frame_out* is high during the entire time of a MAC packet
3. *dclk_out* only transitions when valid data is generated. That is, the *dclk_out* signal should stay low between frames. The rising edge of *dclk_out* should be at the center of the data bits.
4. Data consists of 7 bytes of preamble, 1 byte of SFD, 1 byte of Destination Address, 1 byte of Source Address, 1 byte of PDU Length, N bytes of data where N is the PDU Length, and 1 byte of FCS.
5. Preamble of *data_out* begins with '1' when *frame_out* rises and has a 1 0 1 0 1 0 … pattern for 7 bytes.
6. The SFD has the pattern of 1 0 1 0 1 0 1 1 and occurs after the last 0 of the preamble. That is, it continues the alternating pattern of 1s and 0s of the preamble until its last bit, which is a 1.
7. Data in the MAC packet has same values as the data read from the *LLC_PDU_Gen* Buffer RAM. First check that the data on the output of the *packet_sr_data* MUX is the same as *data_in* during the data portion of the frame. Next, with the *MAC_packet_sr* expanded so you see the individual bits of the shift register, verify that the data from *packet_sr_data* is correctly loaded into the *MAC_packet_sr* and then shifted out of the register, LSB first. Finally, verify that you can see the correct byte patterns appearing in the serial *data_out* with the LSB appearing first.
8. The waveform "state" should sequence through the states 1 through 8 and back to 1.
9. Byte_count should count the bytes in the preamble and PDU data fields.
10. Crc_frame_in should be active during states 4 through 7.

- **Submit TWO images of the simulation waveforms**

    1. The first waveform image shall show two full MAC frames.
    2. The second waveform image shall zoom in to the smaller of the two frames such that the values of the data

    Both images shall show at least the following signals for the *MAC Frame Generator*:

data_in[7:0]	load_packet_sr	sel_dest_addr
data_ready	MAC_packet_sr[7:0]	sel_source_addr
data_ack	shift_count[2:0]	sel_length
pdu_length[7:0]	byte_count[5:0]	sel_pdu_data
dclk_out	state	enable_crc
data_out	en_shift_ctr	
frame_out	en_byte_ctr	
CRC_frame_in	sel_preamble	
packet_sr_data[7:0]	sel_sfd	

    Set the radix of data_in and packet_sr_data to hexadecimal and the radix of pdu_length, shift_count, and bye_count to unsigned decimal.

- **Submit the following code file:** Mac_Frame_Gen.vhd

PART C – Manchester Encoder/Decoder Design

Design Hierarchy/Files

Test_Lab7C_md – behavior (Test_Lab7C_md.vhd)
    uut – manchdecoder – md_beh (md.vhd)

Test_Lab7C – behavior (Test_Lab7C.vhd)
    uut – Lab7C_top_level – Behavioral (Lab7C_top_level.vhd)
        RST_debounce – debounce – Behavioral (real_debounce.vhd)
        ManEncoder – manchencoder – me_beh (me.vhd)
        ManDecoder – manchdecoder – md_beh (md.vhd)

For this part of the lab, the Manchester encoder and decoder designs from Lab 3 will be modified. In Lab 3, the Manchester encoder needed to generate a signal with a '1' level when data was not being transmitted. The decoder only needed to maintain sync with the data for one character (11 bits). For Ethernet, the channel needs to be "quiet" (normally '0') when no data is being transmitted and for longer frames requires that the decoder be designed to maintain sync throughout the entire frame.

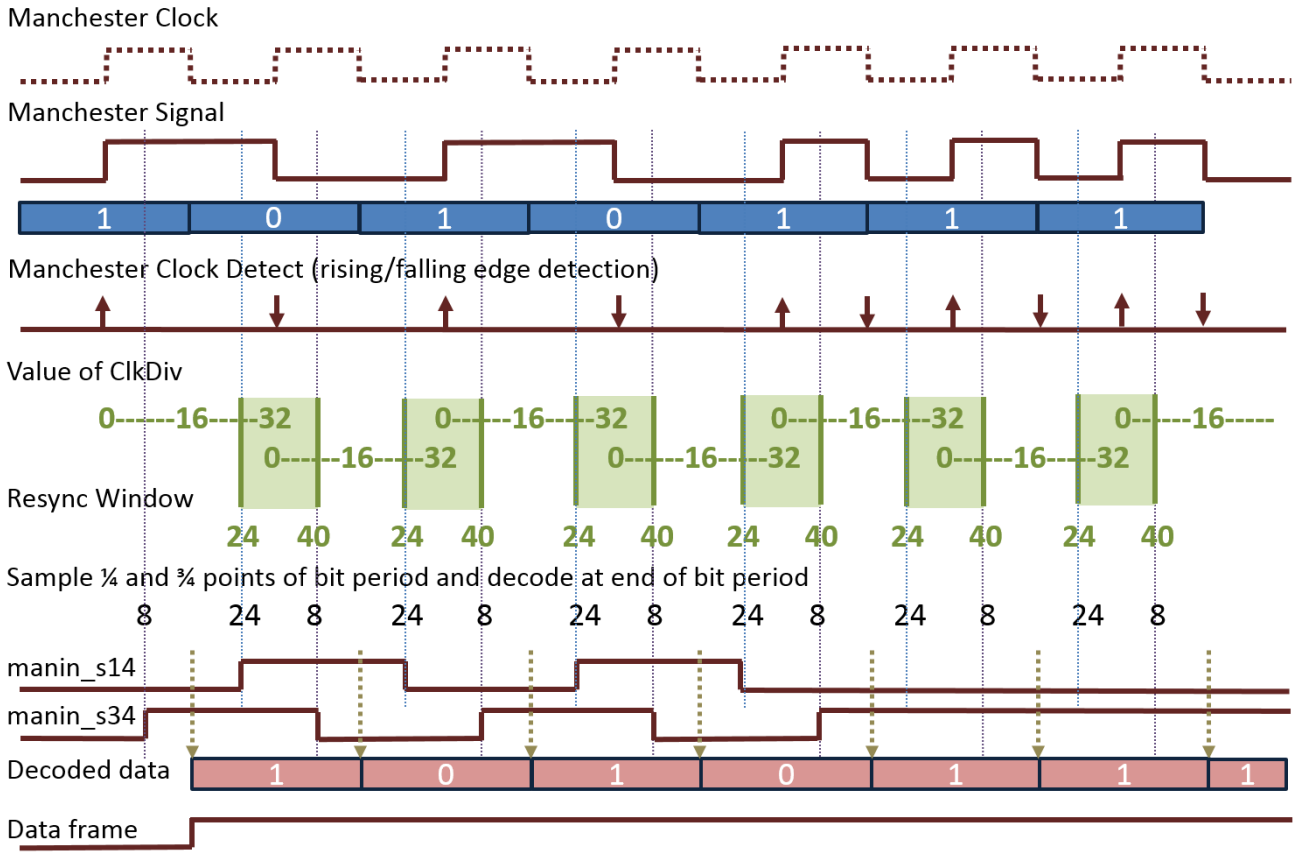
For the Manchester Encoder, you will add a data clock input, dclk_in, and a frame input, frame_in. The data clock will set the data rate of the NRZ input data. The frame input will be a high level during a data frame. The output of the encoder will stay at a '0' level when there is no NRZ input data (i.e. frame input is '0'). The Manchester output is to be synchronized to the data clock input (i.e. the data input clock is the Manchester clock). All of this will simplify the encoder to a single VHDL process.

The Manchester Decoder is to be modified to maintain synchronization through the entire frame of data. When the Ethernet bus is idle, the signal will be steady state at '0'. When a signal is transmitted, the decoder will first expect the MAC preamble pattern (101010…). The clkdiv counter is to sync to the first rising edge it detects, and then to subsequent expected clock edges (+ or -). The period of a data bit is assumed to be 32 system clock periods. The decoder will simply decode 0's and 1's and no longer will search for a "start" bit as in Lab3. Furthermore, the decoder will not search for the start of frame delimiter (SFD). That will be done later by the MAC frame decoder. The Manchester decoder will generate data clock (dclk_out) and frame (frame_out) output signals along with the NRZ data (data_out) decoded from the Manchester signal. Normal operation is depicted in the timing diagram below.

Synchronization to the Manchester signal will be maintained by resetting the clkdiv counter to zero if a Manchester signal transition is detected when the next clock transition is expected. This would be at a counter value of 32, established by clkdiv as the period of one bit of Manchester data. To allow for differences in clock rates, we need to check for the transition to occur each side of 32. So, we establish a window where if clkdiv is in the range of 24 to 40 when a transition is detected, it is accepted and the clkdiv counter is cleared. This window allows for the transition to be up to ±25% of a data-bit period from the expected time. If no transition is detected in the window, the counter will continue to count up to 63 and roll over to 0. Synchronization will then be lost. The decoder will attempt to resync as long as there are transitions on the signal. However, errors have already been introduced into the frame, and ultimately the frame will likely be discarded.

The decoder design is simulated separately so that it can be tested with data rates that are not synchronous to the system clock. The Test_Lab7C_md.vhd test bench provides that capability. A second test bench, Test_Lab7C.vhd, allows the encoder and decoder to be tested together.

Manchester Decoder



Part C Procedure:

- **Continue using your project named "Lab7" in the ISE Project Navigator**
- **Add the provided Lab7 Part C VHDL files to your project**
- **Complete the modifications to the encoder and decoder modules**
  1. The provided files me.vhd and md.vhd are templates that provide guidance.
  2. Use your code from Lab 3 as needed.
- **Run "Synthesize - XST" on the top level module (Lab7C_top_level) to check for Errors and Warnings**

  There should be no errors or warnings generated when running synthesis. If you have error or warning messages, make the appropriate corrections to your VHDL code.
- **Use simulation to verify correct operation of the Manchester decoder**
  1. Use the provided test bench file Test_Lab7C_md.vhd.
  2. Run the test bench for 120 microseconds.

3. The test bench will verify that the decoder operates with +10% and -10% variations in the input data rate. It uses a variable to set the clock period of the Manchester clock. The decoder is tested with the test data pattern with three different values of the variable: nominal Manchester clock period, nominal – 10%, and nominal + 10%.

Verify that the waveform view shows three frames of 32 data bits each:

- 1st has Manchester clock period of 1000 ns,
- 2nd has Manchester clock period of 900 ns, and
- 3rd has Manchester clock period of 1100 ns.

Thus, the durations of the three frames should be: **32, ~28.8, ~35.2 microseconds**, respectively.

4. The test bench has assert statements that will generate warning messages in the simulator console if the nrz_out data is different than expected.

   Verify that there are no error messages in the iSim console pane generated by the test bench assert statements. The assert statements check the NRZ_out to have the following data pattern for each of the three frames:

   10101010 10101011 00001111 00110011

   If there are no NRZ data errors, the iSim console output should look similar to the following:

   ISim>
   # run 120.00us
   Simulator is doing circuit initialization process.
   Finished circuit initialization process.
   at 5 us: Note: Manchester clock period = nominal value (/test_lab7c_md/).
   at 6078125 ps(3): Note: testing for bad NRZ bit values (/test_lab7c_md/).
   at 38609375 ps(2): Note: NO ERRORS FOUND (/test_lab7c_md/).
   at 39500 ns: Note: Manchester clock period = 10% low (/test_lab7c_md/).
   at 43671875 ps(3): Note: testing for bad NRZ bit values (/test_lab7c_md/).
   at 73109375 ps(2): Note: NO ERRORS FOUND (/test_lab7c_md/).
   at 73950 ns: Note: Manchester clock period = 10% high (/test_lab7c_md/).
   at 78578125 ps(3): Note: testing for bad NRZ bit values (/test_lab7c_md/).
   at 114203125 ps(2): Note: NO ERRORS FOUND (/test_lab7c_md/).
   ISim>

5. Verify that dclk_out rising edges occur in the center of each bit period of NRZ_out.
6. Verify that frame_out "frames" the decoded NRZ_out data.

- **In the Lab assignment, submit a PDF of the simulation waveforms for the Test_Lab7C_md test bench**

The view is to include the three frames with at least the following test bench signals:

- rst
- manin
- dclk_out
- frame_out
- nrz_out
- tb_man_clk
- check_nrz

- **Use simulation to verify correct operation of the Manchester decoder together with the encoder**

    1. Use the provided test bench file Test_Lab7C.vhd.
    2. Run the test bench for 40 microseconds.
    3. This test bench will also check for nrz_out data errors and will generate error messages in the simulator console if the nrz_out data is different than expected.

- **Demonstrate the operation of your encoder and decoder to a lab proctor for check off**

Evaluation Criteria for Simulation with the Test_Lab7C.vhd test bench:

1. dclk_out, frame_out, and NRZ_out should look just like dclk_in, frame_in, and NRZ_in, respectively, except with some delay.
2. There are no error messages in the iSim console pane generated by the test bench assert statements. The assert statements check the NRZ_out to have the following data pattern:

    10101010 10101011 00001111 00110011

    If there are no NRZ data errors, the iSim console output should look similar to the following:

    ISim>
    # run 40.00us
    Simulator is doing circuit initialization process.
    Finished circuit initialization process.
    at 3109375 ps(3): Note: testing for bad NRZ bit values (/test_lab7c/).
    at 35671875 ps(2): Note: NO ERRORS FOUND (/test_lab7c/).
    ISim>

- **In the Lab assignment, submit a PDF of the simulation waveforms for the Test_Lab7C test bench**

The simulation duration is to be 40 microseconds with at least the following test bench signals:

- rst
- dclk_in
- frame_in
- nrz_in
- manout
- manin
- dclk_out
- frame_out
- nrz_out
- data_byte[7:0]

- **Submit the following code files:** me.vhd and md.vhd

PART D – MAC Frame Decoder Design

Design Hierarchy/Files

Test_Lab7D – behavior (Test_Lab7D.vhd)

uut – Lab7D_top_level – Behavioral (Lab7D_top_level.vhd)

RST_debounce – debounce – Behavioral (real_debounce.vhd)

Gen_PDU_debounce – debounce – Behavioral (real_debounce.vhd)

LLC_PDU_GenA – LLC_PDU_Gen – Behavioral (LLC_PDU_Gen.vhd)

LLC_Data_Buffer – Data_Buffer – Behavioral (Data_Buffer .vhd)

MAC_Frame_GenA – MAC_Frame_Gen – Behavioral (MAC_Frame_Gen.vhd)

FCS_Gen – CRC – Behavioral (CRC.vhd)

EncoderA – manchencoder – me_beh (me.vhd)

DecoderB – manchdecoder – md_beh (md.vhd)

MAC_Frame_DecB – MAC_Frame_Dec – Behavioral (MAC_Frame_Dec.vhd)

CRC_Error_Det – CRC – Behavioral (CRC.vhd)

HexDisp – HEXon7segDisp – Behavioral (HEXon7segDisp.vhd)

This part of the lab adds the MAC Frame Decoder to the system and integrates the Manchester encoder and decoder modules. A template for MAC_Frame_Dec.vhd is provided. Your CRC.vhd file developed in Lab 4 will be used again in this module. The top level and test bench source are provided as well as the source for the HexDisp. The latter is modified from that used in our previous labs. In this application, if an error is detected, the second decimal point from the right on the display will blink.

Module Description

A block diagram for the MAC Frame Decoder is shown below. As input data is shifted into the data shift register, the state machine waits for the SFD data pattern to appear in the shift register. Thus, the preamble and SFD fields are “stripped” off and not saved. Then the source and destination address fields are captured into registers as well as the PDU length field. The destination address is provided on the dest_addr_out port of the module so that we can compare it with the intended destination address in the top level module.

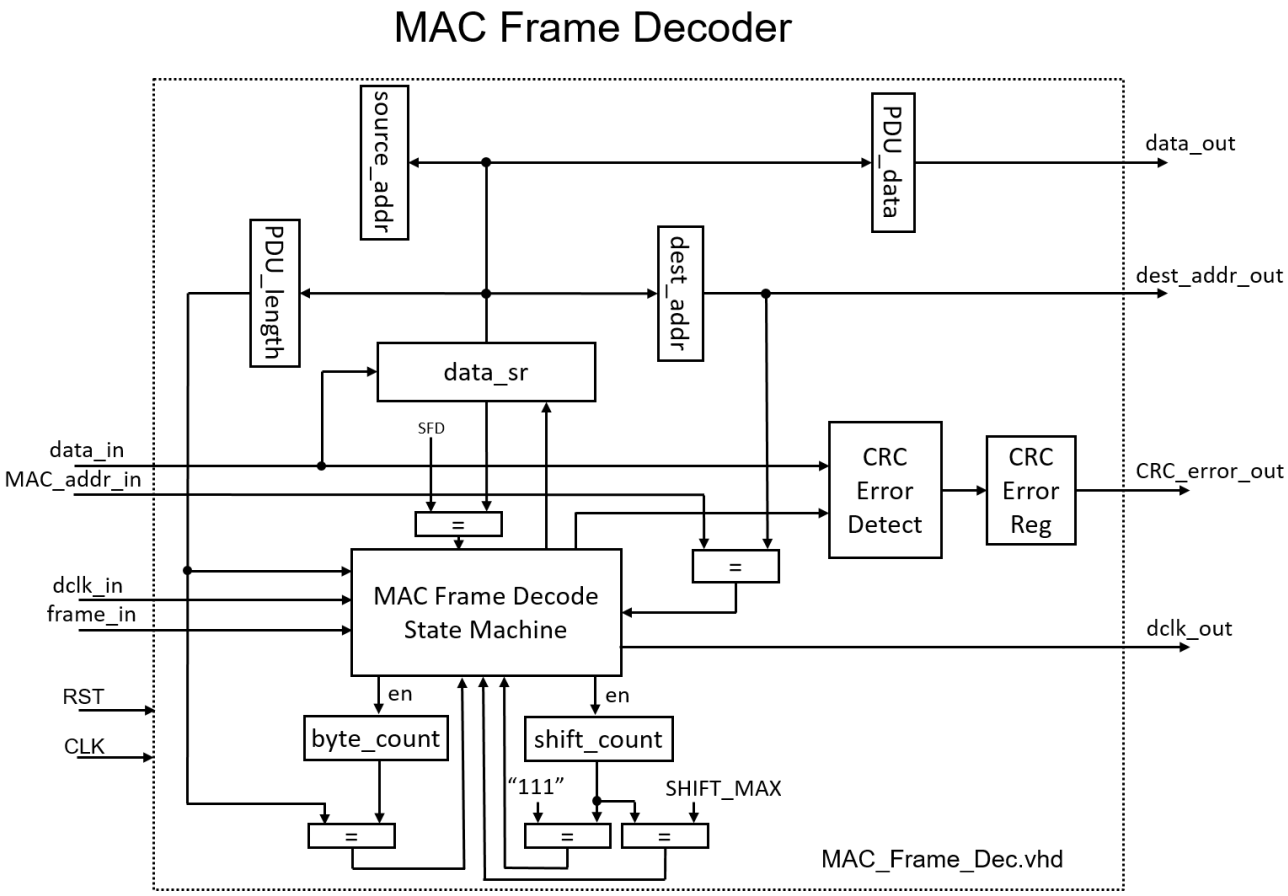
As each byte of the data field of the frame is received, it is captured in a register. The output of this register is immediately presented on the 8-bit data_out port of this module and dclk_out provides a clock output in sync with the data_out. A register in the top level module successively captures each byte of the data. Using the value of the PDU length field to count the bytes of data, when the last byte of the data field is received, the dclk_out stops and the last byte is captured into the register and held. Normally, all of the data would be captured and then sent to the LLC layer on the receiving end. But, to keep our implementation simple, we just capture the last byte of the data field so that we can use it to verify if the data communication was successful.

While the frame is being shifted in and parsed, it is also fed into the CRC module which performs error checking on the frame (not including the preamble and SFD fields). If an error is detected, the CRC_error_out output is activated.

The MAC address input is used to determine if the received data frame is intended for this system on the network by comparing it with the destination address in the received frame. If the received

destination address does not match the MAC address, then the process is terminated and the frame is “discarded”.

The module outputs include the data captured from the last byte of the PDU data, the destination address, the CRC error output, and a data clock to indicate that the frame has been received and the outputs contain valid data for the frame.



Part D Procedure:

- **Continue using your project named “Lab7” in the ISE Project Navigator**
- **Add the provided Lab7D VHDL files to your project**
- **Complete the design of the MAC Frame Decoder module**.
- **<u>Draw a state diagram showing each state of the state machine, the inputs used for transition decisions, and the outputs produced</u> (Submit as a file in the lab assignment or a hardcopy to the TA)**
- **Run “Synthesize - XST” on the top level module (Lab7D_top_level) to check for Errors and Warnings**

  There should be no errors and only the warnings included below when running synthesis.

	Synthesis Messages - Errors, Warnings, and Infos
INFO	Xst:3210 - "C:\Users\Carl\Documents\EECE 359 - Computer Networks\LABS - PAPILIO DUO\Lab7 - Ethernet MAC Protocol\Source Code\MAC_Frame_Gen.vhd" line 149: Output port <error_out> of the instance <FCS_Gen> is unconnected or connected to loadless signal.
WARNING	Xst:647 - Input <P<0:0>> is never used. This port will be preserved and left unconnected if it belongs to a top-level block or it belongs to a sub-block and the hierarchy of this sub-block is preserved.
WARNING	Xst:647 - Input <P<8:8>> is never used. This port will be preserved and left unconnected if it belongs to a top-level block or it belongs to a sub-block and the hierarchy of this sub-block is preserved.
INFO	Xst:3210 - " MAC_Frame_Dec.vhd" line 113: Output port <data_out> of the instance <CRC_Error_Det> is unconnected or connected to loadless signal.
INFO	Xst:3210 - " MAC_Frame_Dec.vhd" line 113: Output port <dclk_out> of the instance <CRC_Error_Det> is unconnected or connected to loadless signal.
INFO	Xst:3210 - " MAC_Frame_Dec.vhd" line 113: Output port <frame_out> of the instance <CRC_Error_Det> is unconnected or connected to loadless signal.
INFO	Xst:3218 - HDL ADVISOR - The RAM <Mram_RegFile> will be implemented on LUTs either because you have described an asynchronous read or because of currently unsupported block RAM features. If you have described an asynchronous read, making it synchronous would allow you to take advantage of available block RAM resources, for optimized device usage and improved timings. Please refer to your documentation for coding guidelines.
WARNING	Xst:1293 - FF/Latch <PDU_length_reg_6> has a constant value of 0 in block <LLC_PDU_Gen>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <PDU_length_reg_7> has a constant value of 0 in block <LLC_PDU_Gen>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1293 - FF/Latch <MAC_Frame_GenA/FCS_Gen/state_FSM_FFd1> has a constant value of 0 in block <Lab7D_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:2677 - Node <MAC_Frame_GenA/FCS_Gen/error_out> of sequential type is unconnected in block <Lab7D_top_level>.
WARNING	Xst:2677 - Node <MAC_Frame_DecB/CRC_Error_Det/data_in_d2> of sequential type is unconnected in block <Lab7D_top_level>.
WARNING	Xst:2677 - Node <MAC_Frame_DecB/CRC_Error_Det/C_reg_out> of sequential type is unconnected in block <Lab7D_top_level>.
WARNING	Xst:1293 - FF/Latch <MAC_Frame_GenA/FCS_Gen/fcs_bit_ctr_5> has a constant value of 0 in block <Lab7D_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <MAC_Frame_GenA/FCS_Gen/fcs_bit_ctr_4> has a constant value of 0 in block <Lab7D_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <MAC_Frame_DecB/CRC_Error_Det/fcs_bit_ctr_0> has a constant value of 0 in block <Lab7D_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <MAC_Frame_DecB/CRC_Error_Det/fcs_bit_ctr_1> has a constant value of 0 in block <Lab7D_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <MAC_Frame_DecB/CRC_Error_Det/fcs_bit_ctr_2> has a constant value of 0 in block <Lab7D_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <MAC_Frame_DecB/CRC_Error_Det/fcs_bit_ctr_3> has a constant value of 0 in block <Lab7D_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <MAC_Frame_DecB/CRC_Error_Det/fcs_bit_ctr_4> has a constant value of 0 in block <Lab7D_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <MAC_Frame_DecB/CRC_Error_Det/fcs_bit_ctr_5> has a constant value of 0 in block <Lab7D_top_level>. This FF/Latch will be trimmed during the optimization process.
INFO	Xst:2261 - The FF/Latch <MAC_Frame_GenA/CRC_data_in_d1> in Unit <Lab7D_top_level> is equivalent to the following FF/Latch, which will be removed : <MAC_Frame_GenA/FCS_Gen/data_in_d1>
INFO	Xst:2261 - The FF/Latch <MAC_Frame_GenA/CRC_data_in_d2> in Unit <Lab7D_top_level> is equivalent to the following FF/Latch, which will be removed : <MAC_Frame_GenA/FCS_Gen/data_in_d2>

- **Use simulation to verify correct operation of the MAC Frame Decoder**

    1. Use the provided test bench file Test_Lab7D.vhd.
    2. Run the test bench for 1560 microseconds.

    The provided test bench will generate a series of "generate PDU" push button actions. Every other time, the switch settings are changed so that the destination address is different than the MAC address for the decoder. This will test for proper termination when the frame is not intended for this receiver.

- **Demonstrate the operation of your MAC Frame Decoder in the simulator to a lab proctor and get checked off**

    Evaluation Criteria for Simulation Results:

    1. For the first MAC packet, the state machine should sequence through the states for a valid data frame, waiting in state 2 until the SFD is decoded and repeating states 9 and 10 for the correct number of data words specified by the PDU length.
    2. For the second MAC packet, the state machine should go into the "terminate" state after the destination address is compared with the local MAC address.
    3. After the first frame finishes, the last_LLC_data_reg[7:0] and the last_MFD_data_reg[7:0] in the top level module should be the same value. After the second frame finishes, these values should be different. After the third frame, they should share the same value again.

- **Submit a PDF of the simulation waveforms to the lab assignment**

    Zoom to just the first TWO frames in the simulation. Include at least all of the following signals:

    UUT
    - last_LLC_data_reg[7:0]
    - last_MFD_data_reg[7:0]

    MAC_Frame_Decoder
    - dclk_in
    - data_in
    - frame_in
    - dclk_out
    - data_out[7:0]
    - mac_addr_in[7:0]
    - dest_addr_out[7:0]
    - crc_error_out
    - data_sr[7:0]
    - shift_count[2:0]
    - byte_count[5:0]
    - pdu_length_reg[7:0]
    - pdu_data_reg[7:0]
    - state
    - en_shift_ctr
    - en_byte_ctr
    - load_dest_addr
    - load_source_addr
    - load_pdu_length
    - load_pdu_data
    - ignore_error
    - crc_frame_in

- **Submit the following code files to the lab assignment:** MAC_Frame_Dec.vhd and CRC.vhd

PART E – System Demonstration on FPGA Board

In this final part of the lab, you will implement your system on your FPGA Board, debug and fix any problems, and then demo your system to a lab proctor.

Procedure:

- **Add the following provided files to your Lab7 project:**
  1. Lab7E_top_level.vhd
  2. HEXon7segDisp.vhd

- **In the ISE Project Navigator Implementation view, select the top level module to be implemented**

  Set **Lab7E_top_level** as your "top module" by doing the following:

  a. Select the "Implementation" Design View
  b. Right click on "Lab7D_top_level" in the Hierarchy
  c. Click "set as top module"

- **Add the provided UCF file, Lab7-PapilioDuo.ucf, to your Lab7 project**

  Note: The previous step (setting of the top module) must be completed before you add the UCF file to the project. Otherwise, the UCF file may not be placed correctly in the hierarchy. If you added the UCF before setting the top module, remove the UCF file and then add it back to the project.

- **Synthesize the top module (Lab7E_top_level) and check for error/warning messages**

  There should be no errors and only the warnings included below when running synthesis.

	Synthesis Messages - Errors, Warnings, and Infos
INFO	Xst:3210 - "C:\Users\Carl\Documents\EECE 359 - Computer Networks\LABS - PAPILIO DUO\Lab7 - Ethernet MAC Protocol\Source Code\MAC_Frame_Gen.vhd" line 149: Output port <error_out> of the instance <FCS_Gen> is unconnected or connected to loadless signal.
WARNING	Xst:647 - Input <P<0:0>> is never used. This port will be preserved and left unconnected if it belongs to a top-level block or it belongs to a sub-block and the hierarchy of this sub-block is preserved.
WARNING	Xst:647 - Input <P<8:8>> is never used. This port will be preserved and left unconnected if it belongs to a top-level block or it belongs to a sub-block and the hierarchy of this sub-block is preserved.
INFO	Xst:3210 - "C:\Users\Carl\Documents\EECE 359 - Computer Networks\LABS - PAPILIO DUO\Lab7 - Ethernet MAC Protocol\Source Code\MAC_Frame_Dec.vhd" line 113: Output port <data_out> of the instance <CRC_Error_Det> is unconnected or connected to loadless signal.
INFO	Xst:3210 - "C:\Users\Carl\Documents\EECE 359 - Computer Networks\LABS - PAPILIO DUO\Lab7 - Ethernet MAC Protocol\Source Code\MAC_Frame_Dec.vhd" line 113: Output port <dclk_out> of the instance <CRC_Error_Det> is unconnected or connected to loadless signal.
INFO	Xst:3210 - "C:\Users\Carl\Documents\EECE 359 - Computer Networks\LABS - PAPILIO DUO\Lab7 - Ethernet MAC Protocol\Source Code\MAC_Frame_Dec.vhd" line 113: Output port <frame_out> of the instance <CRC_Error_Det> is unconnected or connected to loadless signal.
INFO	Xst:3218 - HDL ADVISOR - The RAM <Mram_RegFile> will be implemented on LUTs either because you have described an asynchronous read or because of currently unsupported block RAM features. If you have described an asynchronous read, making it synchronous would allow you to take advantage of available block RAM resources, for optimized device usage and improved timings. Please refer to your documentation for coding guidelines.
INFO	Xst:3218 - HDL ADVISOR - The RAM <Mram_an_out> will be implemented on LUTs either because you have described an asynchronous read or because of currently unsupported block RAM features. If you have described an asynchronous read, making it synchronous would allow you to take advantage of available block RAM resources, for optimized device usage and improved timings. Please refer to your documentation for coding guidelines.

INFO	Xst:3218 - HDL ADVISOR - The RAM <Mram_seg_out> will be implemented on LUTs either because you have described an asynchronous read or because of currently unsupported block RAM features. If you have described an asynchronous read, making it synchronous would allow you to take advantage of available block RAM resources, for optimized device usage and improved timings. Please refer to your documentation for coding guidelines.
WARNING	Xst:1293 - FF/Latch <PDU_length_reg_6> has a constant value of 0 in block <LLC_PDU_Gen>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <PDU_length_reg_7> has a constant value of 0 in block <LLC_PDU_Gen>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1293 - FF/Latch <MAC_Frame_GenA/FCS_Gen/state_FSM_FFd1> has a constant value of 0 in block <Lab7E_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:2677 - Node <MAC_Frame_GenA/FCS_Gen/error_out> of sequential type is unconnected in block <Lab7E_top_level>.
WARNING	Xst:2677 - Node <MAC_Frame_DecB/CRC_Error_Det/data_in_d2> of sequential type is unconnected in block <Lab7E_top_level>.
WARNING	Xst:2677 - Node <MAC_Frame_DecB/CRC_Error_Det/C_reg_out> of sequential type is unconnected in block <Lab7E_top_level>.
WARNING	Xst:1293 - FF/Latch <MAC_Frame_GenA/FCS_Gen/fcs_bit_ctr_5> has a constant value of 0 in block <Lab7E_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <MAC_Frame_GenA/FCS_Gen/fcs_bit_ctr_4> has a constant value of 0 in block <Lab7E_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <MAC_Frame_DecB/CRC_Error_Det/fcs_bit_ctr_0> has a constant value of 0 in block <Lab7E_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <MAC_Frame_DecB/CRC_Error_Det/fcs_bit_ctr_1> has a constant value of 0 in block <Lab7E_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <MAC_Frame_DecB/CRC_Error_Det/fcs_bit_ctr_2> has a constant value of 0 in block <Lab7E_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <MAC_Frame_DecB/CRC_Error_Det/fcs_bit_ctr_3> has a constant value of 0 in block <Lab7E_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <MAC_Frame_DecB/CRC_Error_Det/fcs_bit_ctr_4> has a constant value of 0 in block <Lab7E_top_level>. This FF/Latch will be trimmed during the optimization process.
WARNING	Xst:1896 - Due to other FF/Latch trimming, FF/Latch <MAC_Frame_DecB/CRC_Error_Det/fcs_bit_ctr_5> has a constant value of 0 in block <Lab7E_top_level>. This FF/Latch will be trimmed during the optimization process.
INFO	Xst:2261 - The FF/Latch <MAC_Frame_GenA/CRC_data_in_d1> in Unit <Lab7E_top_level> is equivalent to the following FF/Latch, which will be removed : <MAC_Frame_GenA/FCS_Gen/data_in_d1>
INFO	Xst:2261 - The FF/Latch <MAC_Frame_GenA/CRC_data_in_d2> in Unit <Lab7E_top_level> is equivalent to the following FF/Latch, which will be removed : <MAC_Frame_GenA/FCS_Gen/data_in_d2>
INFO	Xst:2261 - The FF/Latch <MAC_Frame_GenA/clkdiv_0> in Unit <Lab7E_top_level> is equivalent to the following FF/Latch, which will be removed : <HexDisp/Counter_0>
INFO	Xst:2261 - The FF/Latch <MAC_Frame_GenA/clkdiv_1> in Unit <Lab7E_top_level> is equivalent to the following FF/Latch, which will be removed : <HexDisp/Counter_1>
INFO	Xst:2261 - The FF/Latch <MAC_Frame_GenA/clkdiv_2> in Unit <Lab7E_top_level> is equivalent to the following FF/Latch, which will be removed : <HexDisp/Counter_2>
INFO	Xst:2261 - The FF/Latch <MAC_Frame_GenA/clkdiv_3> in Unit <Lab7E_top_level> is equivalent to the following FF/Latch, which will be removed : <HexDisp/Counter_3>
INFO	Xst:2261 - The FF/Latch <MAC_Frame_GenA/clkdiv_4> in Unit <Lab7E_top_level> is equivalent to the following FF/Latch, which will be removed : <HexDisp/Counter_4>

- **Implement the top module**

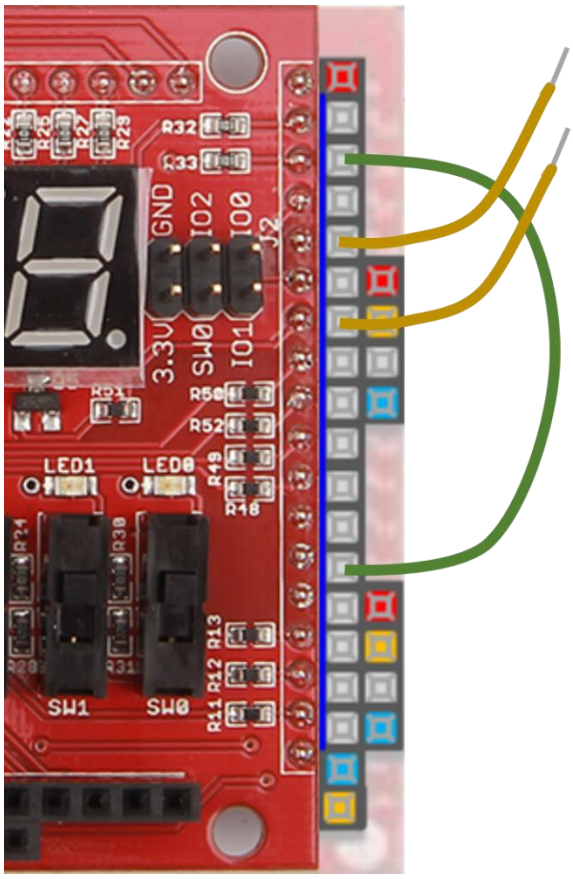
  There should be no implementation warnings. The translate, map, and place & route processes should all have green check marks.

- **Generate the programming file**

  There should be no warning messages for this process.

- **Prepare your FPGA Board**

  For the Papilio Duo with the LogicStart Shield, connect D1 with D11 using a jumper wire. See the figure below.



5V		
D0		
D1	manout	
D2		
D3	MAC_frame	
D4		5V
D5	MAC_data	3.3V
D6		
D7		GND
D8		
D9		
D10		
D11	manin	
D12		5V
D13		3.3V
D14		
D15		GND
GND		
3.3V		

- **Download the programming file to your FPGA Board**

- **Verify the operation of the top module**

1. Set switches 0-3 to an arbitrary 4-bit address for the MAC address for the MAC frame decoder. This sets the MAC address for the receiving end of the communication.
2. Set switches 4-7 to the same 4-bit address as the MAC address. The address pattern should appear on LEDs 4-7. This will be the destination address in the transmitted frame.
3. Press button 0 (reset) and then button 1 (gen PDU). The MAC address on switches 0-3 should now appear on LED 0-3 and each pair of characters on the display should have a random value that is the same. The left value on the display is the last byte of data in the MAC frame that is generated. The right value is the last byte of data received by the MAC frame decoder.
4. Each time you press button 1 (gen PDU), you should see a new pair of random numbers that are the same.

5. None of the decimal points on the display should be lit. If the second decimal point from the left is blinking, it means an error was detected by the CRC. If this happens, it is likely that there is a problem with your design.
6. Now change the value of the destination address (switches 4-7) and press button 1 (gen PDU). Now only the left display should update. The MAC frame decoder should be discarding the frames since the frame's destination address does not match its MAC address.

- **Use the oscilloscope to view the Manchester data**

1. Jumper the D1_manout pin to the D11_manin pin (should have been done previously).
2. Connect three channels of the scope to the "manout", "MAC_frame", and "MAC_data" outputs of the FPGA board.
3. Trigger the scope using the "MAC_frame" signal and capture a single frame on the screen.
4. Compare the Manchester signal with the MAC_data signal and check that they match.
5. Save a screen image file and include it in your lab assignment.

   To save a PNG image file:
   - Image files can be saved to an external USB storage device. Plug a USB flash drive into the USB connector on the front of the scope
   - Press **[Save/Recall] > Save > Format**; then, turn the Entry knob to select **24-bit image (*.png)**.
   - Press the softkey in the second position and use the Entry knob to navigate to the save location.
   - Press the **Settings** softkey.
     In the File Settings Menu, you have these softkeys and options:
     - **Setup Info** — setup information (vertical, horizontal, trigger, acquisition, math, and display settings) is also saved in a separate file with a TXT extension.
     - **Invert Grat** — the graticule in the image file has a white background instead of the black backgound that appears on- screen.
     - **Palette** — lets you choose between Color or Grayscale images.
   - Finally, press the Press to Save softkey.
     A message indicating whether the save was successful is displayed.
6. **<u>Submit the image of your scope screen shot to the Lab assignment.</u>**

- **<u>Demonstrate your board operation to a lab proctor and get checked off</u>**

- **<u>Submit the following code files to the lab assignment:</u>**

1. Lab7E_top_level.bit (your FPGA load file)
2. Submit any previously submitted code files that changed since first submitted and note the changes in the comment section of this assignment.