USB 2.0 to Ethernet Controller for a SoC Design

Design Review Document

ECE 337

Submission by: November 11, 2017
Lab Section: Wednesday 2:30 p.m.
TA: Anirudh Sivakumar

Prepared by:

Alejandro Orozco
Alexandria Symanski
Vishnu Gopal
Akshay Raj

1. Executive Summary

The need for access to internet has become an integral part of modern life especially in the workplace. WiFi can be unreliable especially at crucial moments during downloads or during conference calls, making access to Ethernet is essential. Most modern laptops, Macs for example, do not have an Ethernet port but are commonly used used in an office setting. USB ports are standard on modern laptops and therefore provide a reliable solution to creating an Ethernet adaptor.

The USB to Ethernet bridge converts incoming packet from either the USB or Ethernet side and converts the packet into a format the USB controller can understand. The USB controller waits for more signals from the host to decide where and when the stored data needs to move. The focus of this design is on the USB receiver, USB transceiver, USB controller, USB FIFOs and the MAC. The design will be interfaced with a PHY chip for data moving on the Ethernet side.

The conversion process requires repeated and similar computation so it is ideal for an ASIC design. The adaptor needs to be small for user convenience and to fit in the USB port so an ASIC is more desirable than a microcontroller.

. It will be necessary to fully develop a USB 2.0 transceiver module as well as a USB data transfer controller module compliant with the industry standards as stated later in the document. Each module entitles a significant amount of planning, coding, and testing in order to successfully meet this criteria. Consequently, each module will have relevant documentation including RTL, FSW, and waveform diagrams as well as independent test benches.

Successful design of the proposed accelerator will require the following resources:

- Reference Standard Cell Simulation Library for Mapped Design Verification
- Reference Standard Cell Technology Library for Final Design Layout Verification
- Verilog HDL Simulation and Design Synthesis Tool Chain

The following document contents will describe:

- Proposed system usage and pin map (Section 2)
- Proposed design architecture. (Section 3)

2. Design Specifications

2.1. System Usage

2.1.1. System Usage Diagram

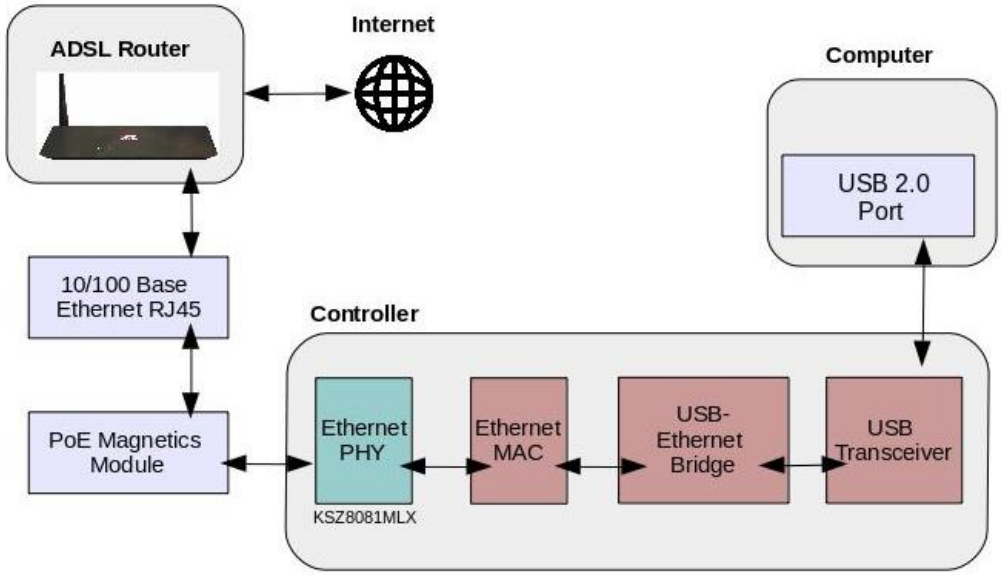


Figure 1: Example System Usage Diagram for USB 2.0 to Ethernet Controller

An example system usage diagram for the controller is described in Figure 1. The system is intended to include the user's USB 2.0 port, USB to ethernet adapter with our embedded controller, PoE magnetic module, an RJ45 cable and a ADSL router. The controller allows ethernet networking over USB, involving application-independent exchange of data in both directions. The RJ45 cable and PoE module serve to transfer data from the controller to the network router similar to a direct connection from an ethernet cable. While the other connection is meant for a USB port of a computer without a direct ethernet port connection. The key operations to be performed by the controller are:

1. Serve as SoC in a USB to ethernet adapter
2. Provide a fast ethernet controller compliant with IEEE 802.3
3. Integrate 10 Mbps Fast Ethernet MAC/PHY
4. Serve as the Physical and Link Layers in an OSI model

2.1.2. Implemented Standards and Algorithms

Standards

- IEEE 802.3 10BASE-T Mbit/s (1.25 MB/s) compatible over twisted pair cable
- IEEE 802.3 Ethernet
- USB Specification Version 2.0
- 10 Mbps Ethernet PHY
- Ethernet Emulation Model (EEM)
- Open Systems Interconnection Model (OSI)

Ports

- USB Type A port
- LAN Ethernet port

Power

- 5V bus (powered by PC)

Ethernet packets are read at a speed of 10Mbit/s. The required clock from PHY is 2.5 MHz for receiving and transmitting. The USB packets are read at high speed mode. The bridge transfers the data by converting Ethernet frames to USB packets and vice versa. The OSI model describes several layers for Ethernet protocol. The first two layers are implemented by this design, that is the physical layer and the link layer. The remaining layers such as the networking and application layers are handled entirely by the software in the user's computer.

2.2. Pinout

Table 1: USB Type A Pinout Description

Signal Name	Direction (IN/OUT/Bidir)	Number of Bits	Description
V	PWR		+5 V
DM	OUT	Serial Out (512 Byte Frame)	Data minus (Receive)
DP	IN	Serial In (512 Byte Frame)	Data plus (Transmit)
GND	GND		Ground

Table 2: Ethernet Bus 10Base-T Pinout (not directly implemented in this design)

Signal Name	Direction (IN/OUT/Bidir)	Number of Bits	Description
Tx+	IN		Transmit Data
Tx-	IN		Transmit Data
Rx+	OUT		Receive Data
NC	No Connection		No Connection
NC	No Connection		No Connection
Rx-	OUT		Receive Data
NC	No Connection		No Connection
NC	No Connection		No Connection

Table 3: Ethernet MAC pinout

Signal Name	Direction (IN/OUT/Bidir)	Number of Bits	Description
TXC	IN		Transmit Clock
TXEN	OUT		Transmit Signal
TXD	OUT	4	Transmit Data
RXC	IN		Receive Clock
RXDV	IN		Receive Signal
RXD	IN	4	Receive Data
RXER	IN		Receive Error
CRS	No Connection		No Connection
COL	IN		Collision Error

2.3 Operational Characteristics

2.3.1 Controller Operations

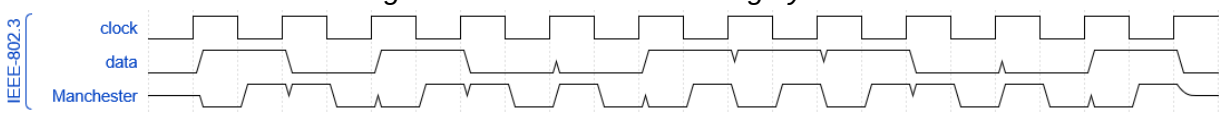
1. USB transfer buffer is loaded
2. USB read pointer is set to beginning of transfer buffer
3. Size of buffer header is read
4. Data in buffer is identified as EEM Command packet or EEM Data packet
5. USB read pointer is moved the size of the packet header (to beginning of packet data)
6. Packet processed based on type
7. USB read pointer is moved the size of the data
8. If the end of the transfer buffer is found, transfer is complete, else start at step 3

It's important to note that an ethernet frame is bigger than the packet size that an USB can process, being that an ethernet frame is usually 1500 bytes and the USB in the design can only process 512 bytes at a time. The ethernet frame is split into packets of a size the USB can process and to notify the USB how many packets make up the frame a command packet is sent first with frame length data. Before each packet is sent a TX command is sent providing information on the following packet. The command holds the data start offset, whether the byte is the first or last segment, and the number of bits or padding in the byte.

2.3.2 MAC and PHY Interface

The Media Access Control block and the Physical Layer Transceiver block is implemented with a standard Media Independent Interface (MII). This design implements the 10BaseT component which follows Manchester coding as a support medium in the 10Mbit/s system as specified by the IEEE-802.3 standards. The encoding done by PHY consists in representing the data bits, '1' and '0', as either a low-to-high or high-to-low as a full cycle on the master clock as illustrated in Figure 2. This encoding is suitable for the 10BaseT standard but would introduce frequency related issues for higher data rate standards such as the 100BaseTx.

Figure 2: Manchester encoding by PHY



The MAC layer transmits and receives bits through MII, which has nine signals. The RXC and TXC are clocks that run at 2.5MHz for a 10Mbit stream. RXDV and TXEN are master signals

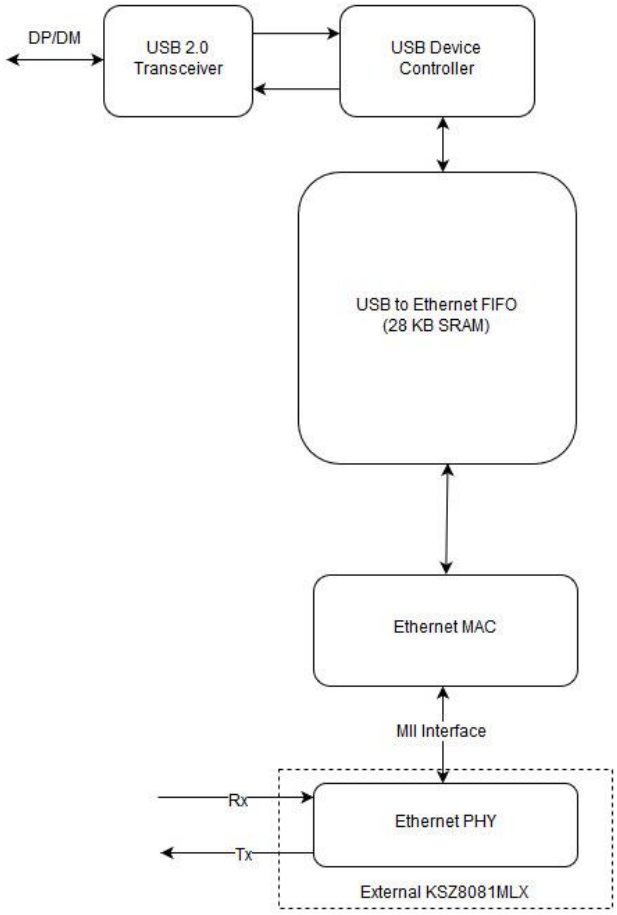
that stay active high throughout the transfer of a packet and the data values are sent in 4 bit streams.

2.4. Requirements for Design

The requirements for the controller are based on the intended target application of providing a USB to Ethernet adapter for office and student use. With this application in mind we have chosen to design a quick and reliable controller with certain optimizations to reduce area. It will be reliant on a USB interface that communicates with a USB host controller that is compliant with USB V2.0. This version of USB is not only backwards compatible but also theoretically gives a transfer speed of 60Mbps, which is sufficient for our usage specification. This will also ensure the greatest possibility that the required drivers will be preloaded. The ethernet controller will implement a 10/100Mbps Ethernet LAN function based on IEEE802.3 standards. This requires a 1.25Mbps signal that can be passed through a coaxial cable. The USB controller will have a single speed mode at this rate and will require a 512 byte memory bus for reading and writing address operations. This speed will accommodate the transfer of a wide range of file types including video and music files. The LAN950x is a high performance solution for USB to 10/100 Ethernet port bridging. The 10/100 Ethernet Controller integrates a MAC and PHY of which we plan to design the MAC. Moreover it is fully compliant with IEEE802.3/802.3u standards. The LAN950x chip contains 56 pins. Ethernet frames are about 1500 bytes in size, or 3 USB 2.0 packets. The controller will need to be able to handle this volume of data, using 3 cycles of operations. This is a better solution as due to design constraints, we would be using a smaller SRAM which will save a lot of area in the design. The USB system will include two kinds of packets. The first one would be Token Packets(To explain what is expected to follow) and Data Packets. The token packets would include IN, OUT, ACK NACK, STALL and DATA which will be assigned a code known as PID for identification. For the USB system, each packet is sent as a single transfer, a series of maximum-length packets ending with a short packet to signal completion. After the transfer there is latency before the next transfer can be sent.

3. Design Architecture

Figure 3: USB to Ethernet Controller Architectural Diagram



The desired architecture implementation is illustrated in Figure 2. The transceiver modules, namely the Ethernet PHY and USB 2.0 PHY, will handle the physical interface as an SoC to manage the data flow. Furthermore, two modules act as device controllers: a USB Device Controller and an integrated Ethernet MAC. The former will handle the USB 2.0 traffic to interface with the FIFO bridge module, and the latter will similarly handle the Ethernet Rx and Tx data to interface with said bridge. Lastly, the FIFO bridge will interface with both controllers and manage the data transfer through the use of an SRAM block for both Rx and Tx data buffering.

3.1 USB Controller

Figure 4: Finite State Machine Diagram for USB Controller

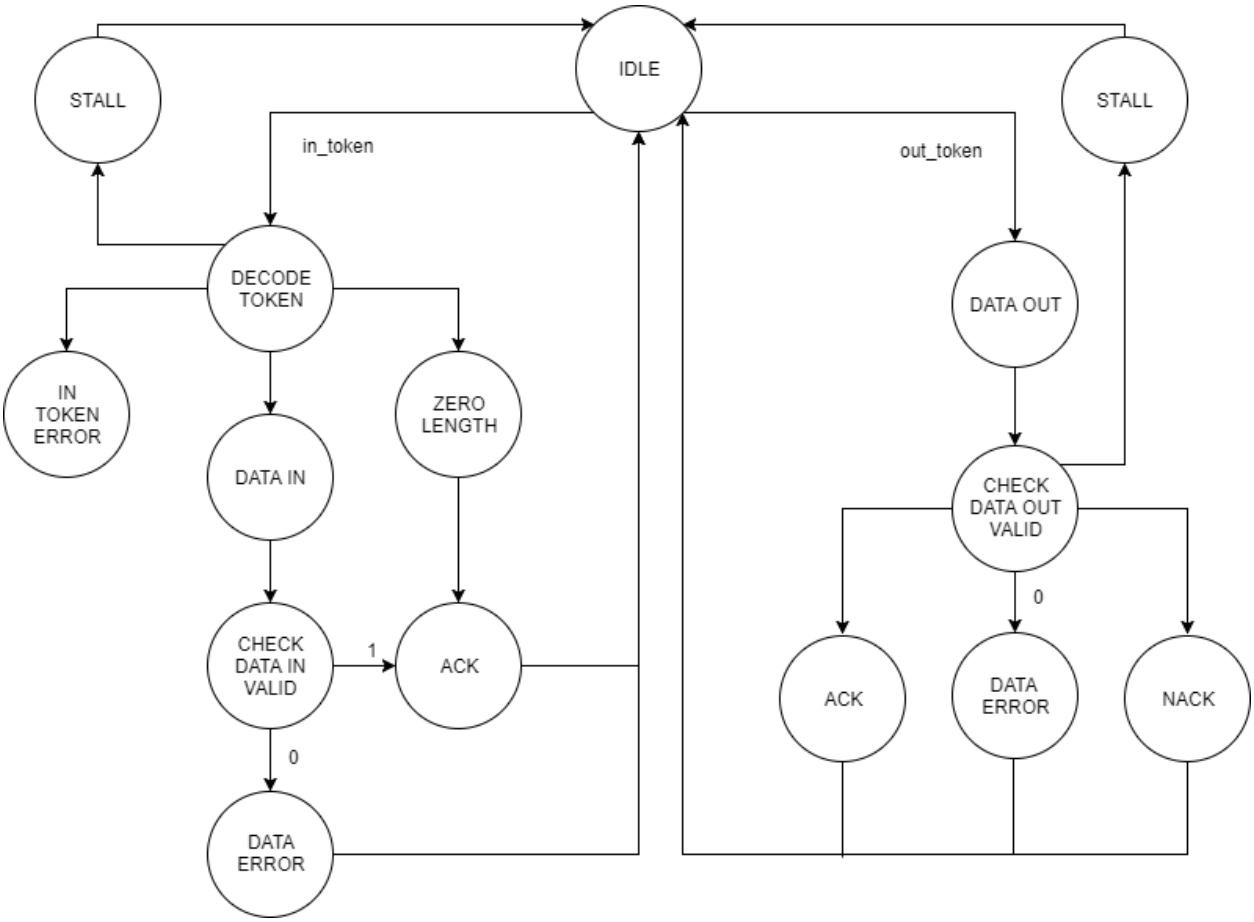
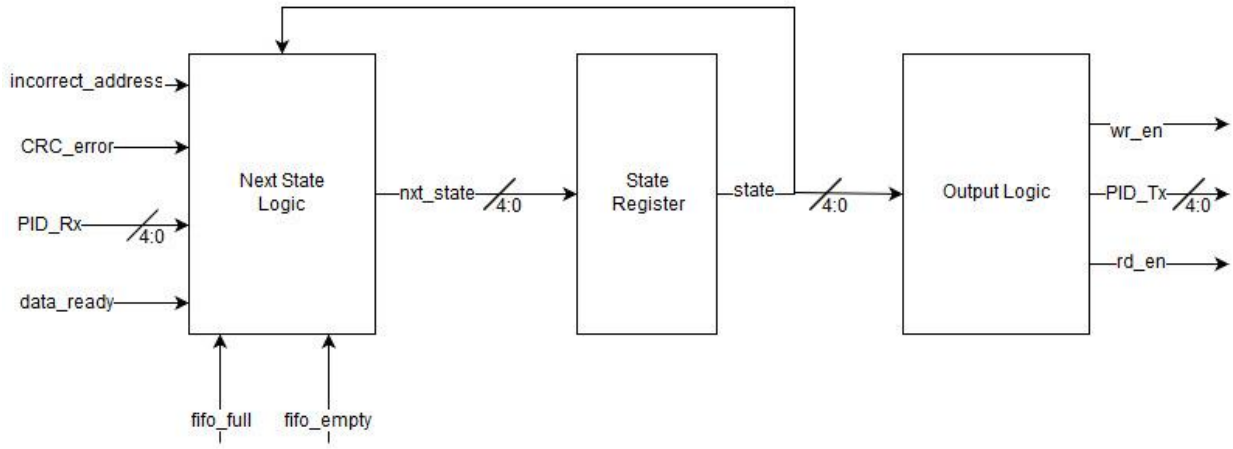


Figure 5: USB Controller RTL



For the bulk data transfer, the host can either send an IN token or an OUT token. If the host wants to receive data, it will send an IN token verified by the receiver that upon validation, the USB controller responds with a data packet from the FIFO. If there's no data in the FIFO, the controller will NAK to indicate that there is no data available, or STALL if an error occurred. If the host wants to transmit data, it will issue an OUT token followed by a payload. The receiver again verifies the token and the incoming data for errors, ignoring the packet if corruption is detected. Otherwise, the USB controller can either NAK if the FIFO is full, ACK if the data was received correctly, or stall if an error occurred.

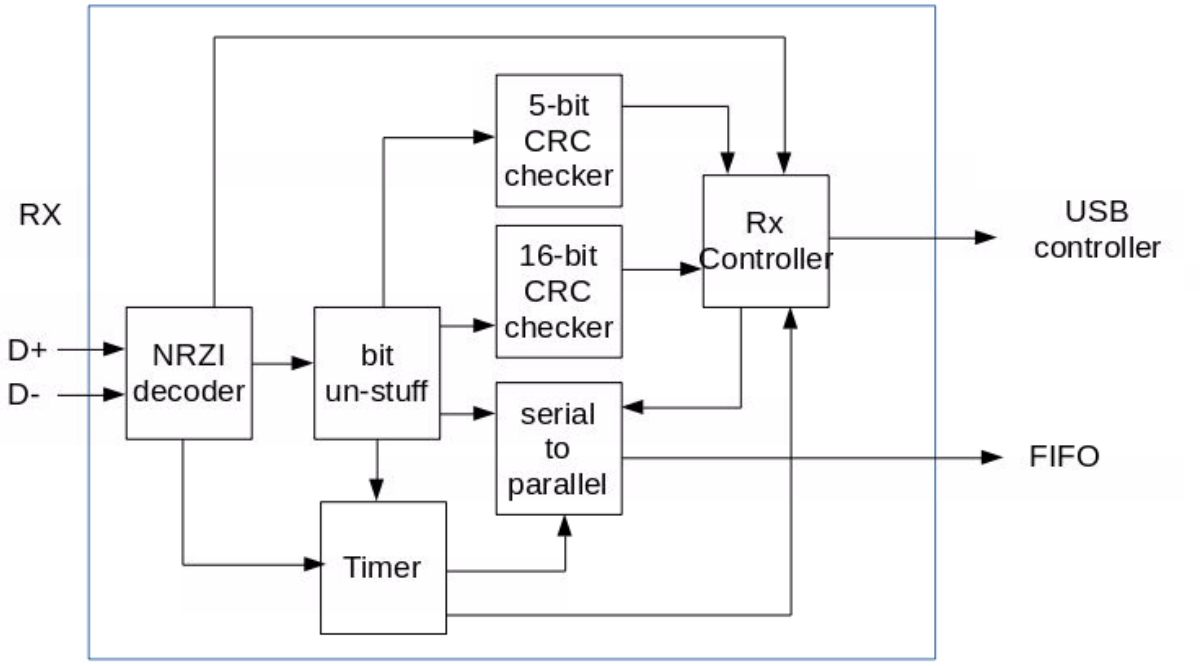
USB device controller area analysis:

Name of Block	Category	Gate/FF Count	Area (um2)	Comments
USB Controller next state	Combinational	9	6,750	Next state logic
USB State Register	Reg. w/ Reset	4	9,600	Assume 15 states

The USB device controller is a simple state machine with 15 states (and thus 4 flip flops as a state register) in addition to an input combinational logic for decoding the PID data type that is being received.

3.2 USB Receiver

Figure 6: USB Receiver Functional Block Diagram



The USB receiver handles the processing of data packets from the host. The data is input into the receiver through D+ and its inverse D-. The data is transmitted through two data streams so that it can be checked for errors and so that the end of packet can be identified when both data streams go low. The incoming data is NRZI (non return to zero, invert on one) encoded so the incoming data must be decoded. NRZI means that the data represents transitions in the data with a one, because it's common for data without transitions for a long period of time, creating a signal of zero volts, bit stuffing is required. Bit stuffing adds a one bit when six non-transitions have been found. The incoming data must be bit-unstuffed for the correct data to be stored in the FIFO. When a bit is being unstuffed a signal is sent to the timer to tell the serial to parallel shift register not to shift in the new bit. Once all the data is in the shift register the controller sends a signal to the USB controller that the data is read to transfer to the FIFO.

Figure 7: Flowchart depicting USB receiver operation

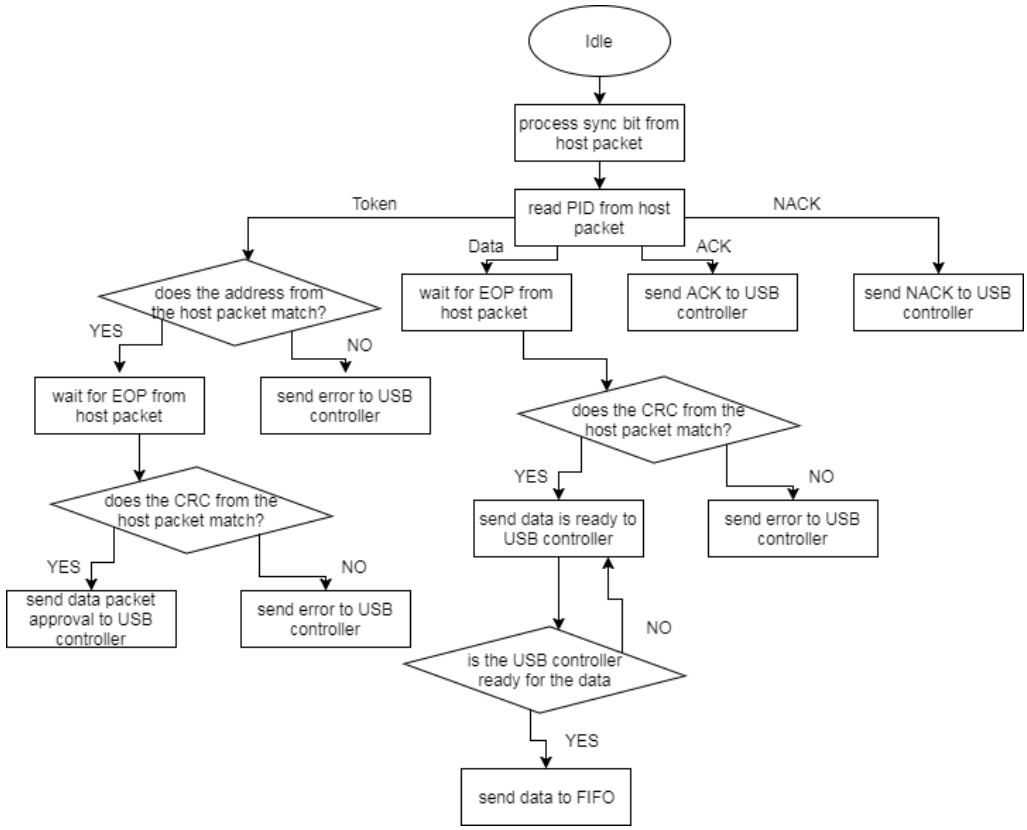
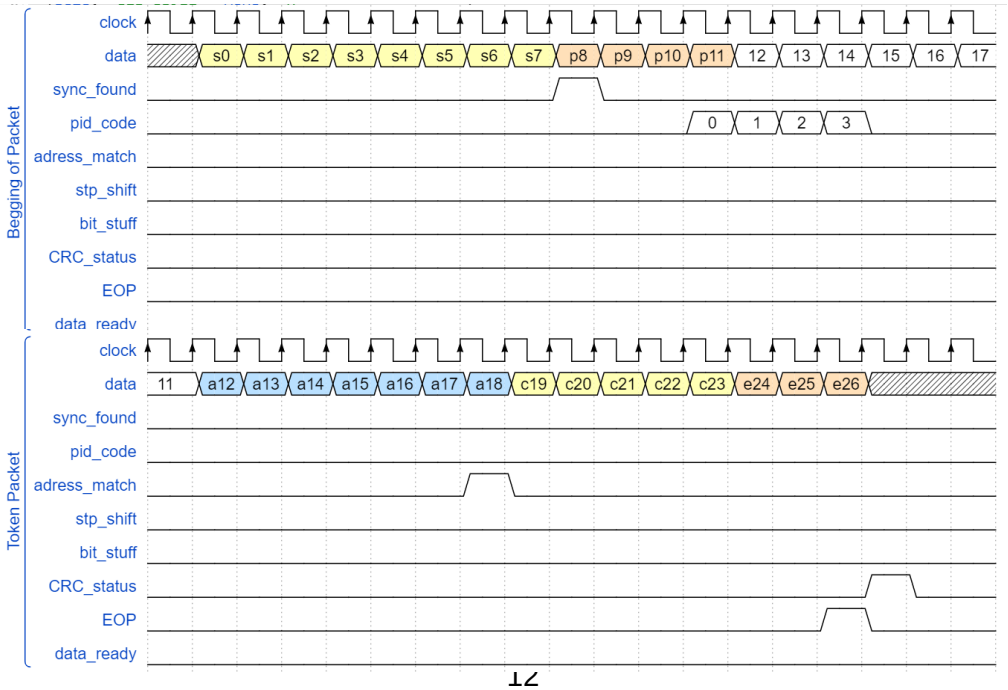
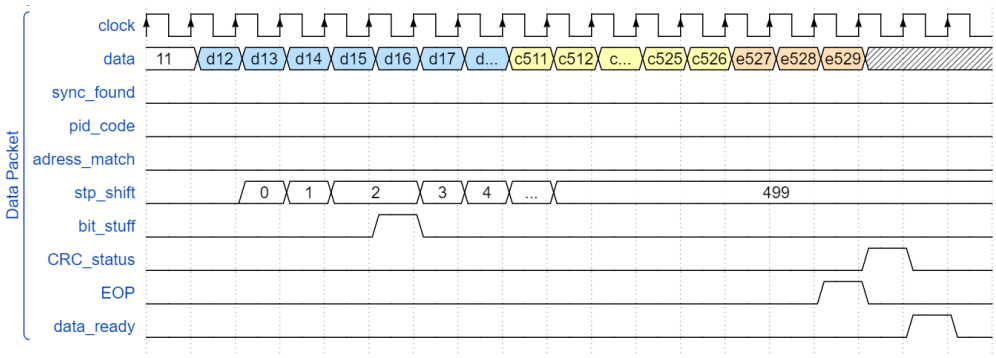


Figure 8: Waveform Diagrams for USB RTL



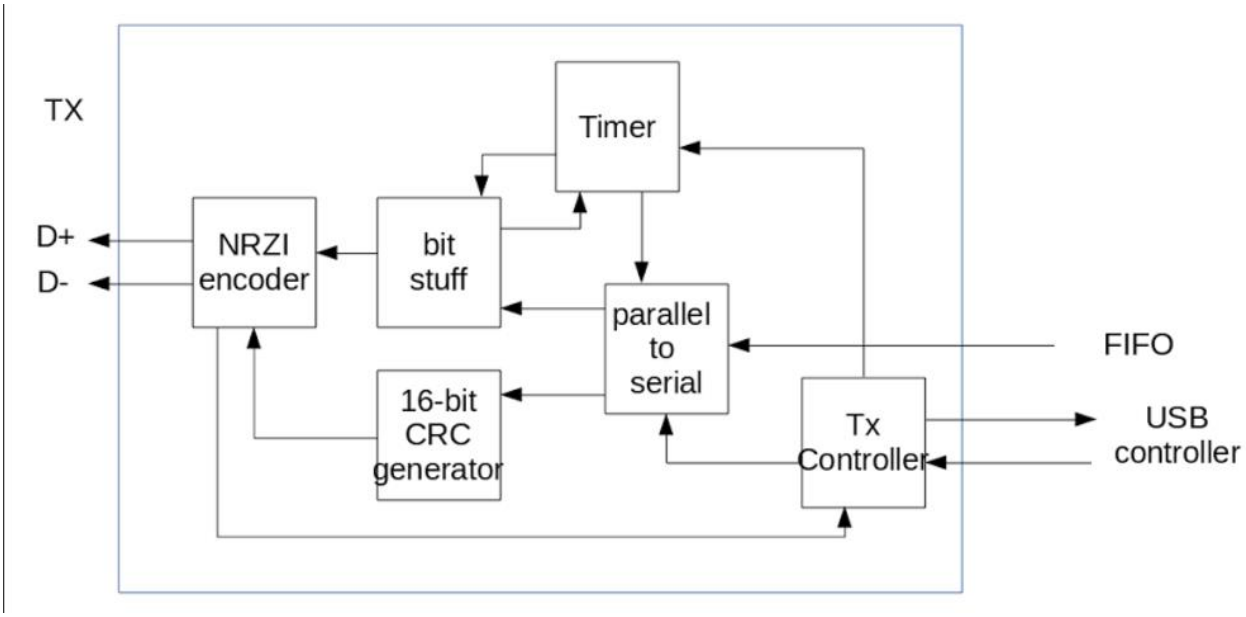


Name of Block	Category	Gate/FF Count	Area (um2)
Receiver controller state reg.	Reg. w/ Reset	4	9,600
Receiver controller next-state	combinational	5	3,750
5-bit CRC checker	Reg. w/ Reset	5	12,000
16-bit CRC checker	Reg. w/ Reset	16	38,400
series to parallel shift register	Reg. w/o Reset	501	676,350
USB receiver timer	Reg. w/ Reset	4	9,600
bit unstuffer	Reg. w/ Reset	4	3,000
NRZI decoder	Reg. w/ Reset	6	14,400

The USB receiver performs some of the core processing in the USB to Ethernet bridge and therefore accounts for a larger part of the overall area (after the FIFO). The biggest component of the receiver area is the serial to parallel because it has 500 registers to store the data.

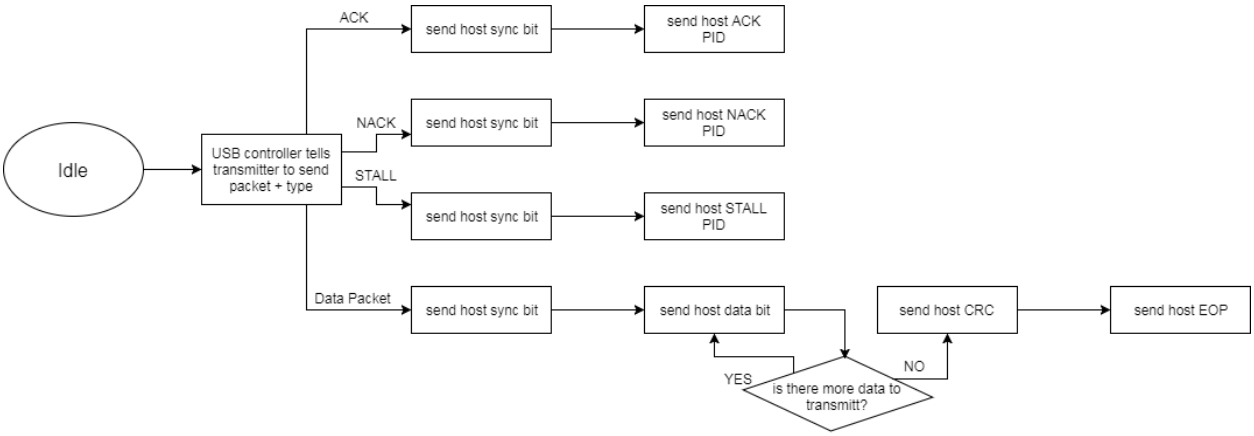
3.3 USB Transmitter

Figure 9: USB Transmitter RTL

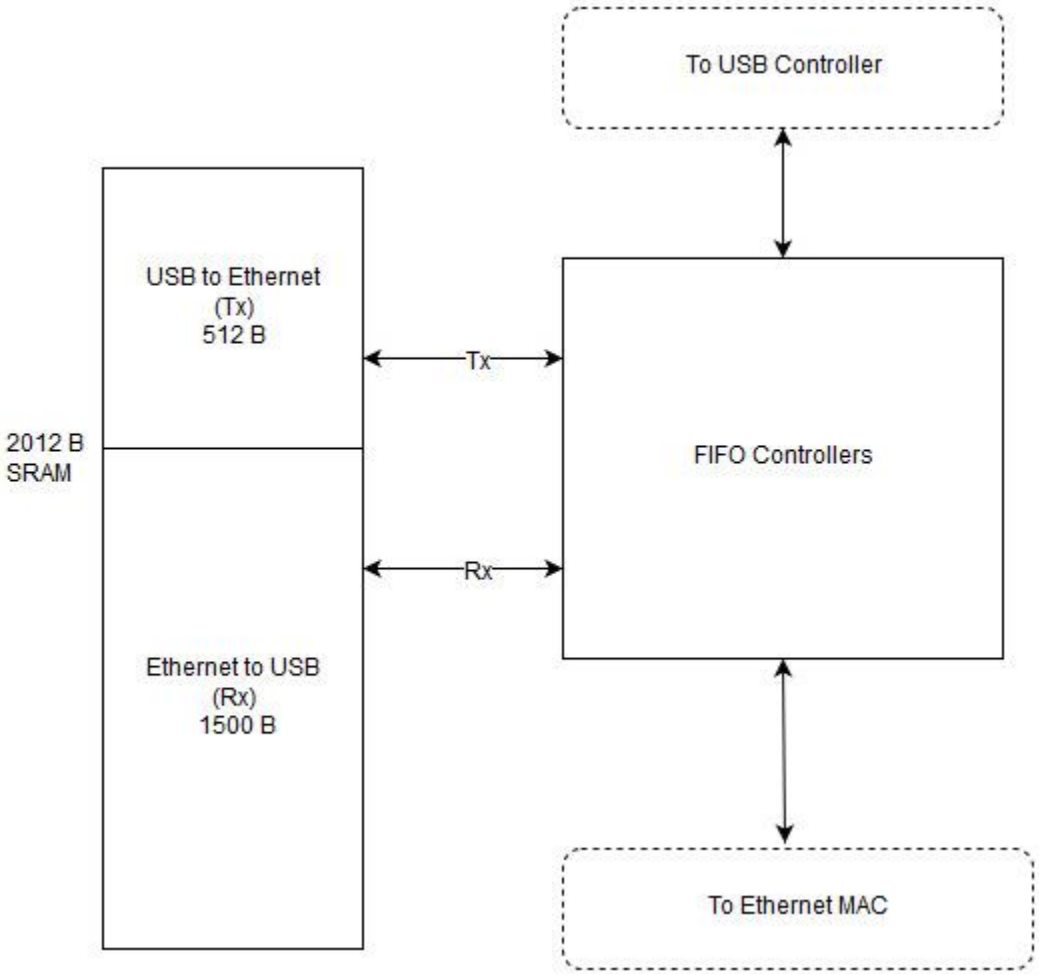


The USB controller communicates with the Tx Controller and requests a handshake packet and type. The packet could be : ACK(to acknowledge data received), NACK(error message), STALL(wait time to complete data transfer that is already processing) and an IN token(to request data). Once the IN token is received the Tx controller informs the timer and parallel to serial(P2S) to get ready for the data transmission. After the data is processed by the P2S, it is bit-stuffed and parallelly, the CRC(Cyclic Rudimentary Check) generates and attaches a 16-bit CRC code at the end of the data packet. This is then NRZI encoded and transmitted out. The NRZI encoder also informs the Tx controller when the data packet transmission is complete. The D+ and D- are inverted versions of each other to check if data is processed correctly.

Figure 10: Flowchart depicting USB transmitter operation



3.4 FIFO



The USB to Ethernet FIFO block is in charge of managing the data transfer between the USB interface and the Ethernet interface. The memory block is composed of two individual FIFO SRAM blocks: a 512 B FIFO to store the USB to Ethernet traffic, and a 1500 FIFO to store the Ethernet to USB traffic. The FIFO's act as buffers and are written to, and read by the respective blocks. The RTL diagrams outlining the sub blocks are illustrated as follows:

Figure 11: USB to Ethernet FIFO RTL Diagram

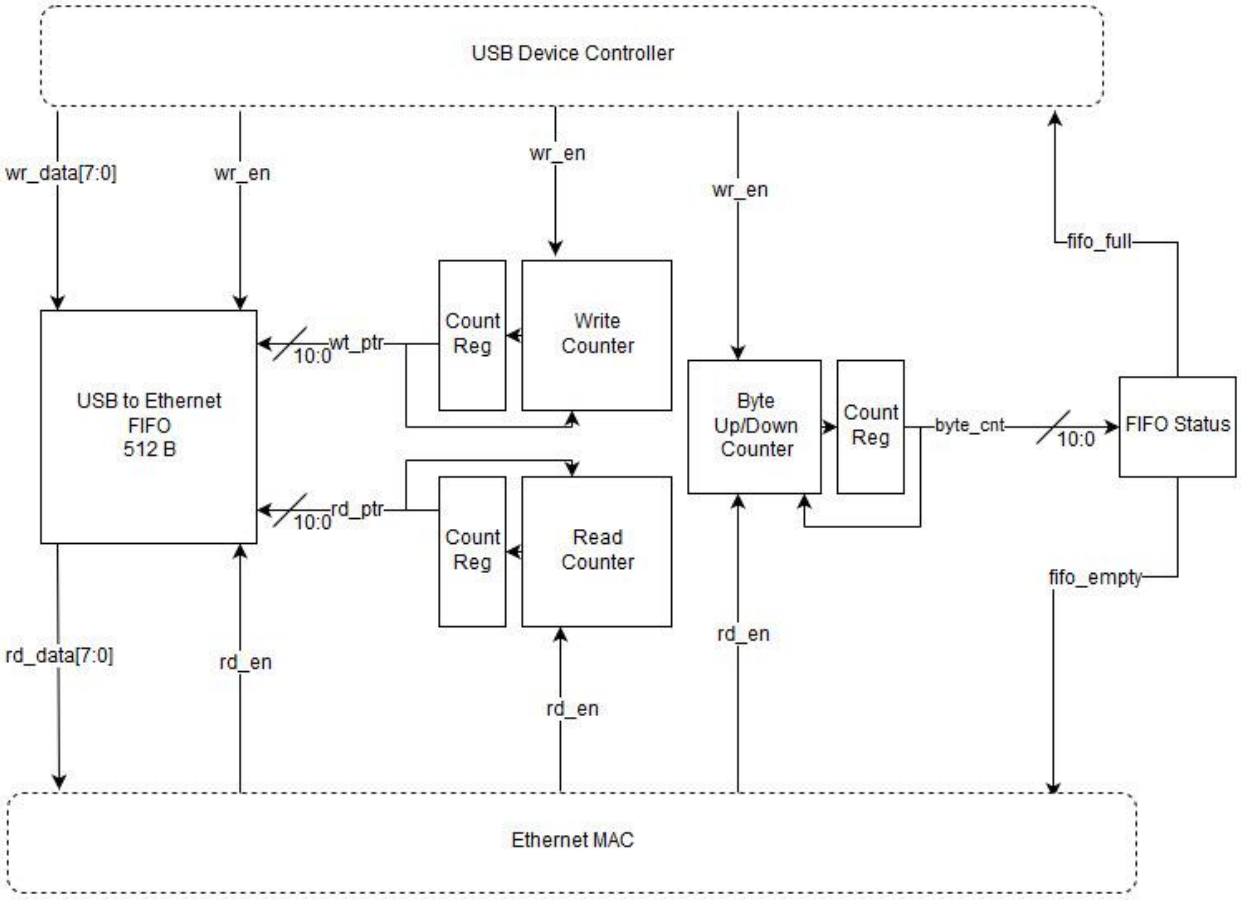
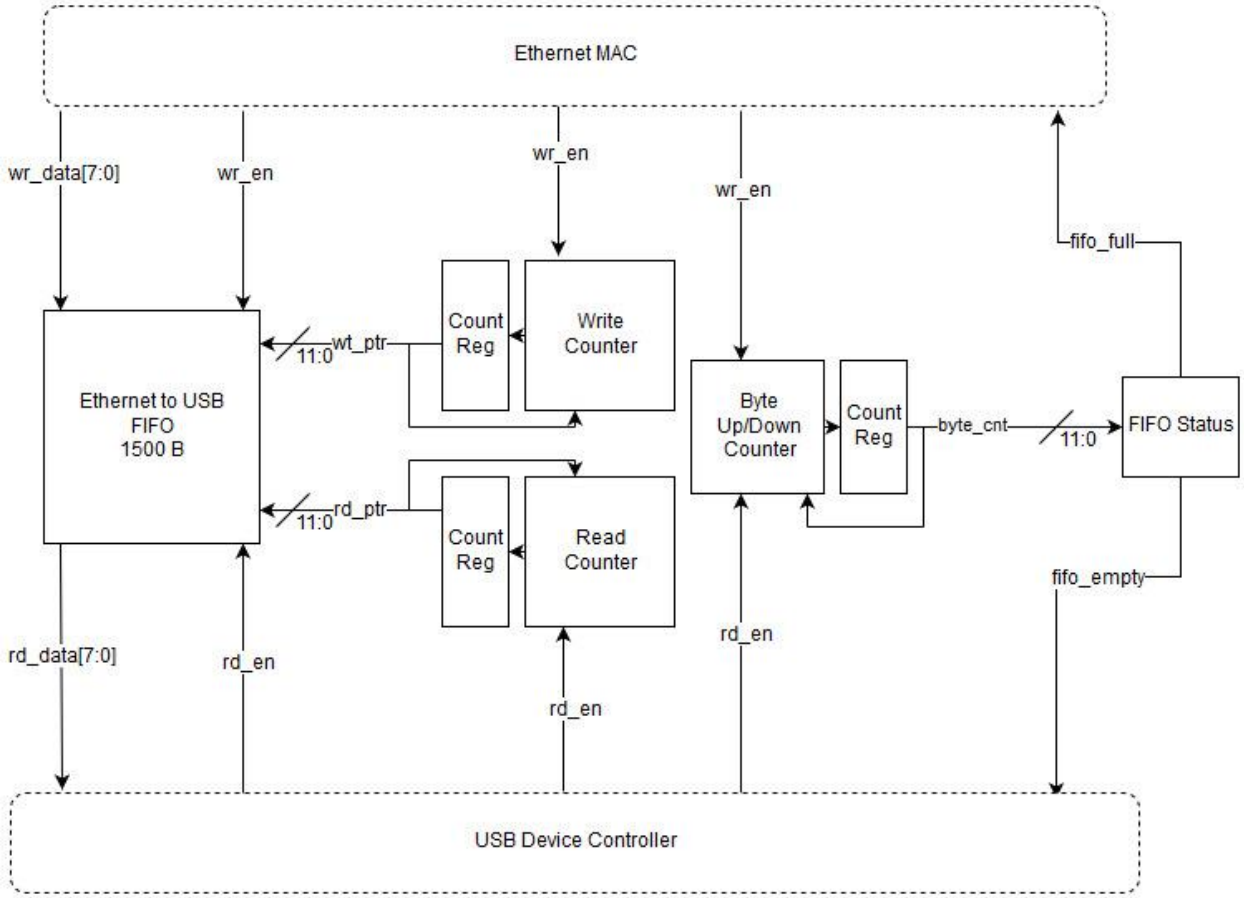


Figure 12: Ethernet to USB FIFO RTL Diagram



The USB to Ethernet and Ethernet to USB FIFOs are essentially two independent instances of the same modules. The FIFO controller is basically three counters: one to keep track of the current write address pointer, one to keep track of the read address pointer, and a third to keep track of the number of bytes currently within the FIFO. The USB to Ethernet and Ethernet to USB FIFOs differ only in the SRAM size (512 B for the former and 1500 B for the latter), and consequently for the width of the bus and the size of the counters that keep track of the address pointers and the number of bytes. The respective parties will be responsible for the write and read enable signals.

FIFO Controller and SRAM area analysis:

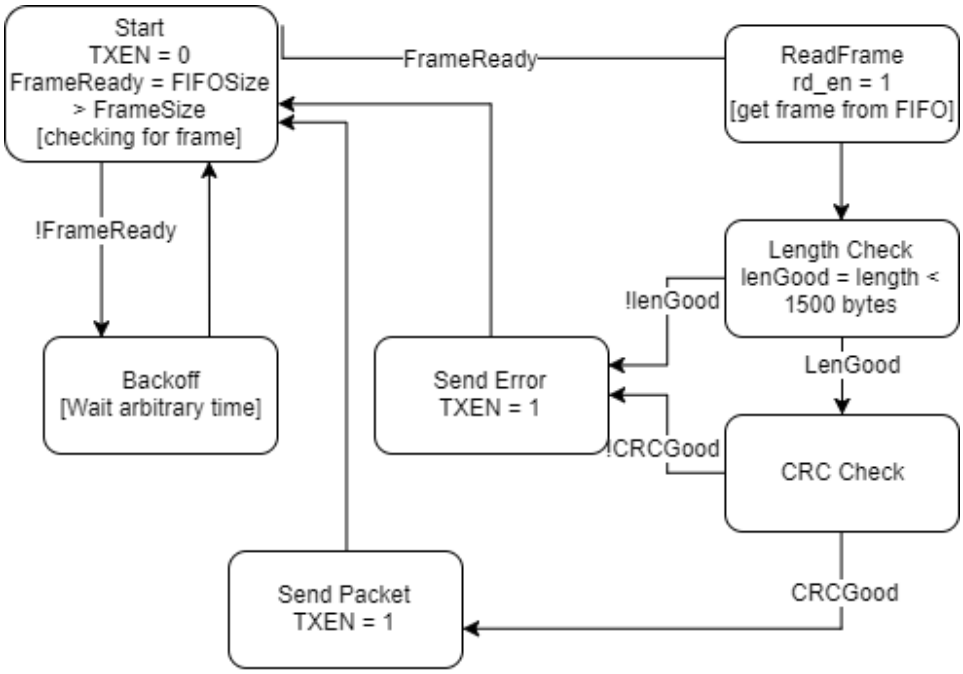
Core Area Calculations				
Name of Block	**Category**	**Gate/ FF Count**	**Area (um2)**	**Comments**
USB to Ethernet FIFO count regs	Reg. w/ Reset	30	72,000	3 10-bit counters
Ethernet to USB FIFO count regs	Reg. w/ Reset	33	79,200	3 11-bit counters
On-chip SRAM	On-chip SRAM	16096	1,207,200	Assumes 2012 B memory
FIFO status	Combinational	23	17,250	a 10-bit and a 11-bit comparator

As mentioned in the block diagrams and RTL diagrams descriptions, the FIFO block of the design incorporates two SRAM blocks and six counters (three for each SRAM block). The total SRAM allocation amounts to 2012 bytes of memory, enough to store the maximum payload size on either direction of transmission (512 bytes for a USB packet and 1500 bytes for an Ethernet packet). Additionally, two sets of three counters each are included in the controllers. Three counters include a 10-bit register to keep track of 512 different addresses and byte count, and the other three include 11-bit registers to keep track the remaining 1500 addresses and byte count of the larger FIFO block. Lastly, two combinational FIFO status block (essentially comparators) is included as a means of explicitly calculating the total amount of data in each FIFO. The FIFO block potentially accounts for the largest amount of area in the entire chip, mainly due to the area allocated for the SRAM blocks despite minimizing them to limit the capacity for 1 maximum size payload on either direction.

3.5 MAC

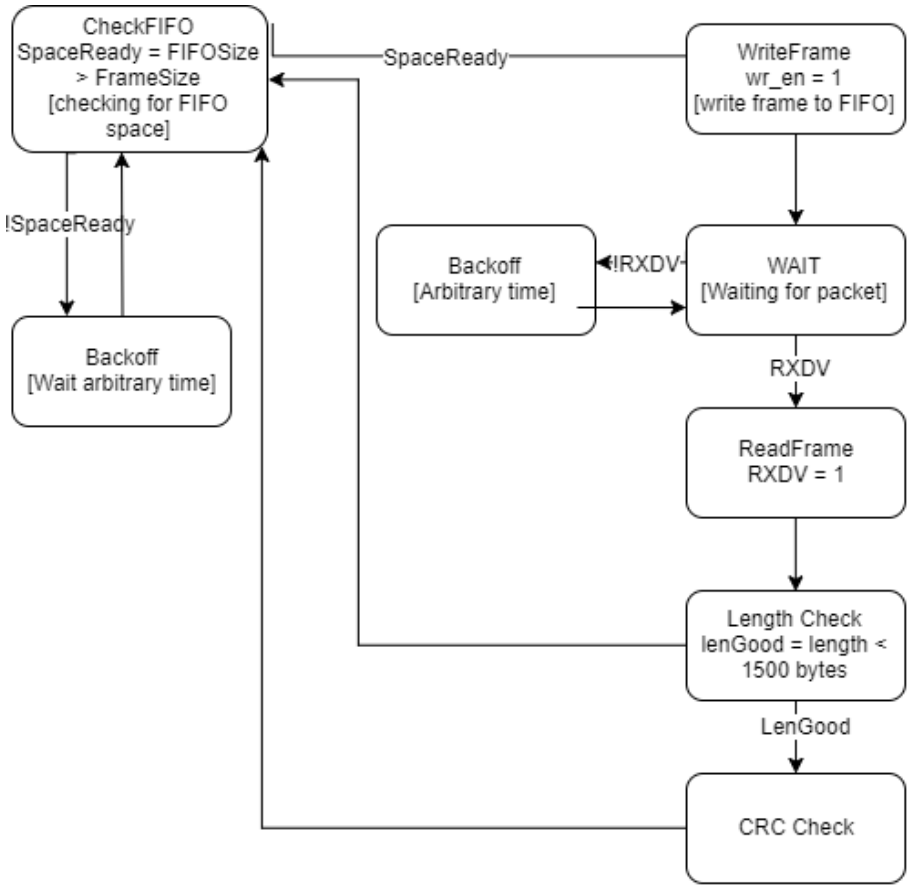
The MAC controller has two independent blocks – one for receiving and one for transmitting data. To transfer data from the FIFO the controller acts similar to the USB controller. First, it checks to see if the FIFO has one frame worth of data to transfer. If it doesn't it waits for an arbitrary fraction of time before checking again. If it does, it enables the rd_en signal and gets the data from the FIFO. It then checks the length of the frame and computes the cyclic redundancy check after filtering out the preamble. If the CRC matches the value stored in the FSC of the frame, and the length is found to be in the correct range, then the controller enable TXEN and the bits are sent to the PHY.

Figure 13: MAC Transmitter State Diagram



The receiver acts in an opposite way to the transmitter. First, it waits for the RXDV signal. Once the RXDV signal is on, the frame is read. Then the CRC and length are checked. If these values match then the data is sent to the FIFO. If not, then an error message is sent.

Figure 14: MAC Receiver State Diagram



The remaining signals in the MII interface between the MAC and the PHY are shown below. The TXC and RXC are the transfer and receive clocks outputted by the PHY. These signals run at a rate of 2.5MHz for a stream rate of 10Mbit/s. The COL signal is enabled when a frame clash occurs on the MAC side. When this signal is enabled an error message will be sent.

Figure 15: MAC-PHY MII Interface

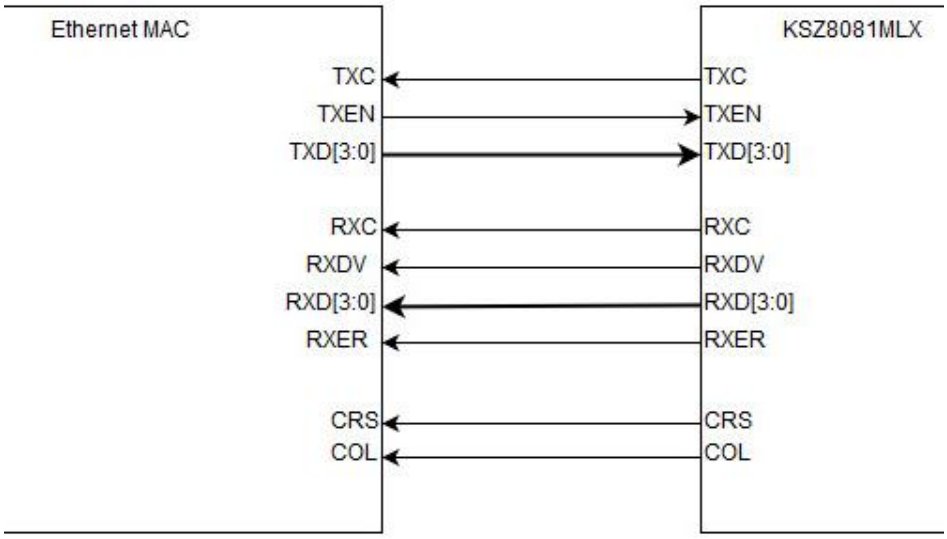
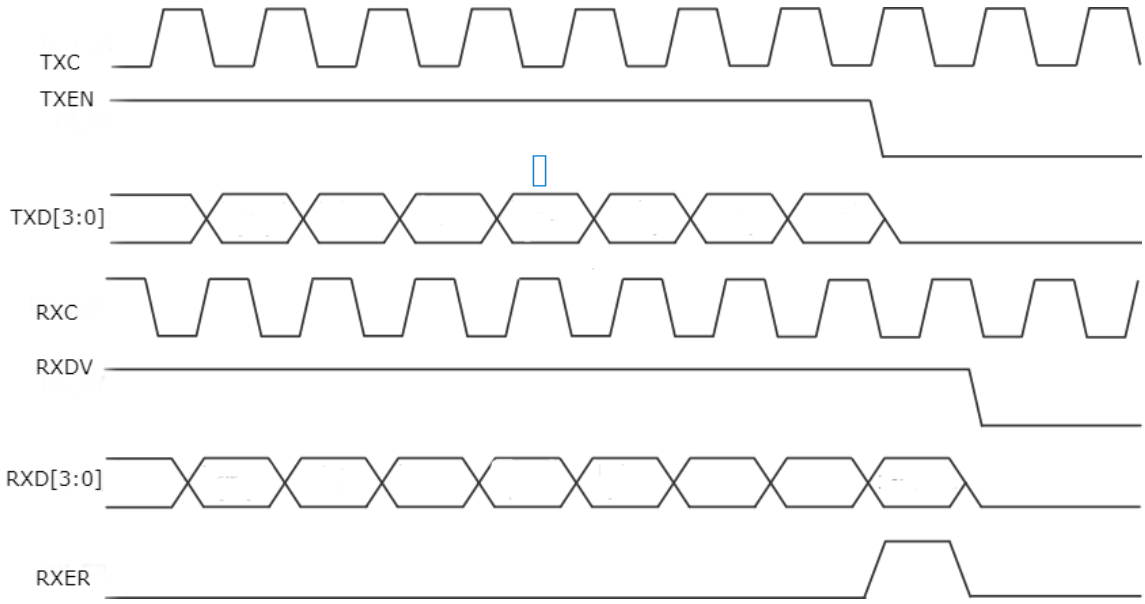


Figure 16: MII Timing waveform



MAC controller area analysis:

Name of Block	Category	Gate/ FF Count	Area (um2)	Comments
MAC Receiver 32-bit CRC checker	Reg. w/ Reset	3	7,200	Estimated area only
MAC Transmitter 32-bit CRC checker	Reg. w/reset	3	7,200	Estimated area only
MAC transmitter state reg	Reg w/reset	4	9,600	Requires at least 4 FF for states
MAC Receiver state reg	Reg w/reset	4	9,600	Requires at least 4 FF for states
MAC transmitter next state reg	combinational	10	7,500	
MAC Receiver next state reg	combinational	9	6,750	

The MAC is a fairly simple block and doesn't require much area to implement. The state registers each have 4 flip-flops and the combinational logic is estimated to have low area. One way to optimize the area of this block would be to implement both receive and transmit blocks in one controller.

3.6 Functional Block RTLs

3.6.1USB Receiver

Figure 17: USB Receiver RTL

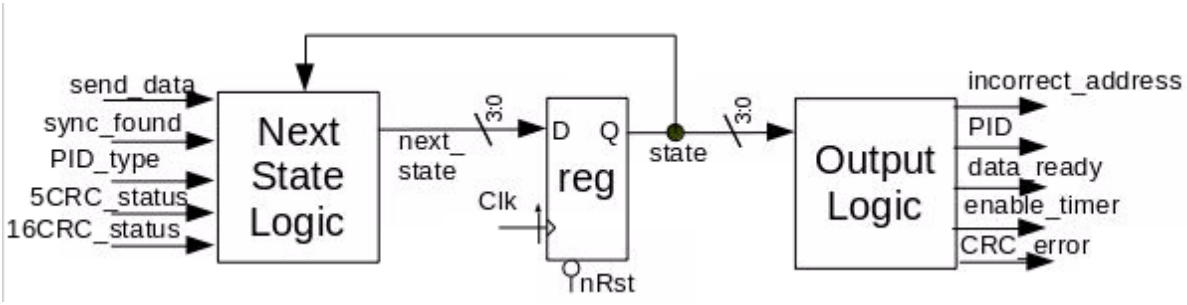


Figure 18: 16-bit CRC RTL

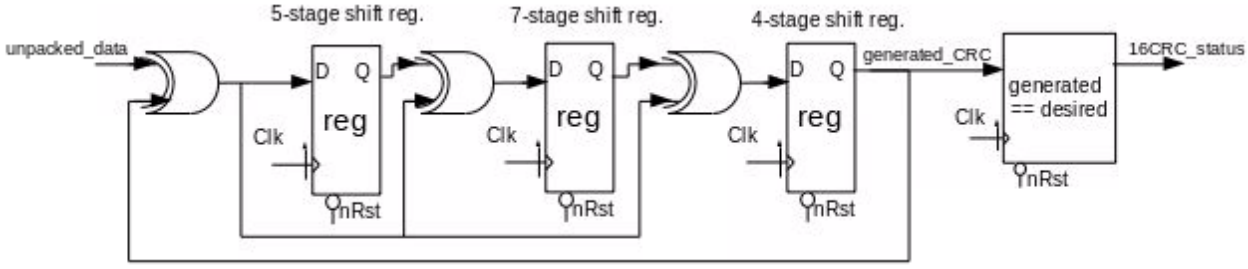


Figure 19: MAC Transmitter RTL

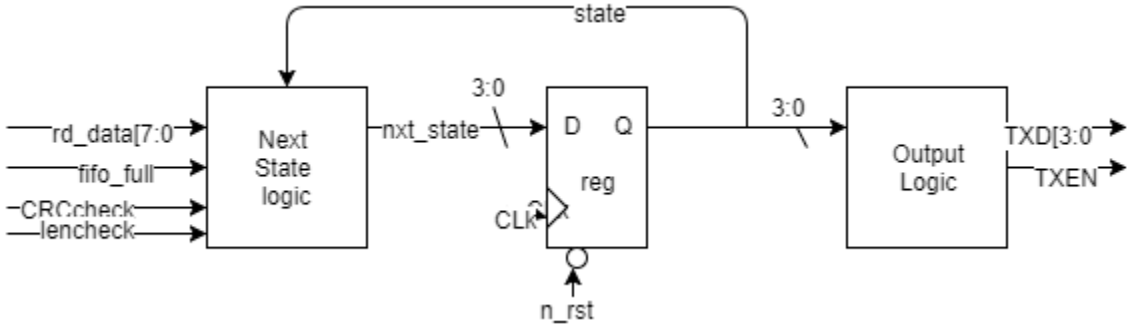


Figure 20: USB Transmitter RTL

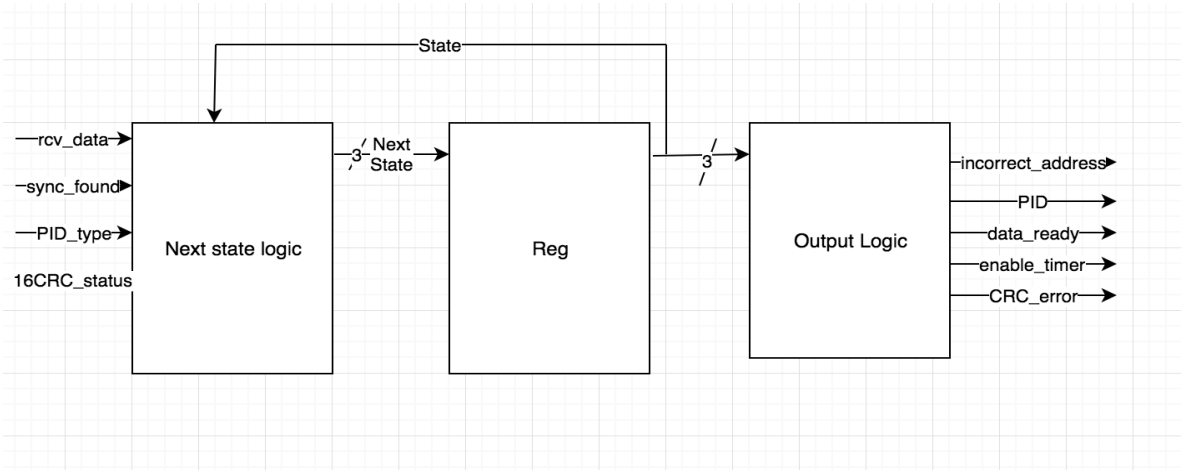


Figure 21 : Cyclic Rudimentary Check RTL

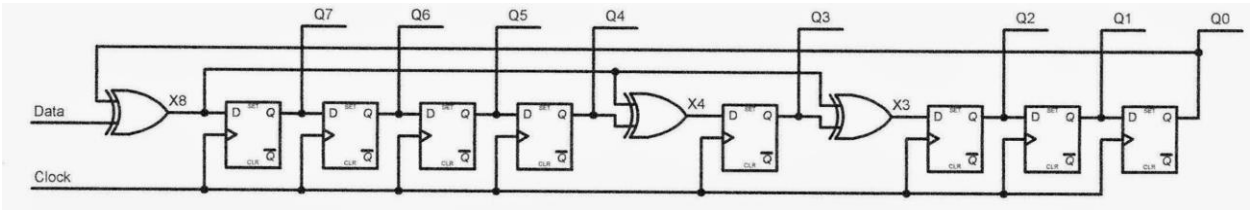


Figure 22: NRZI encoder RTL

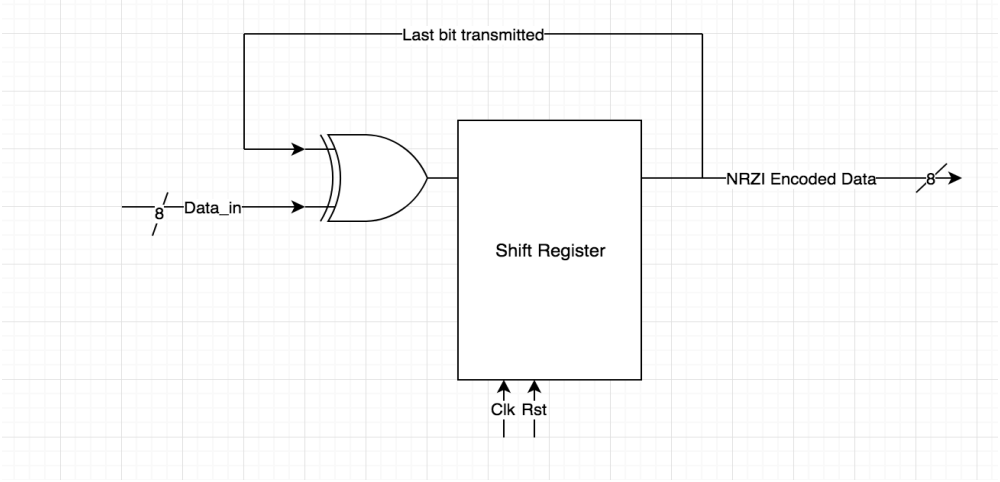
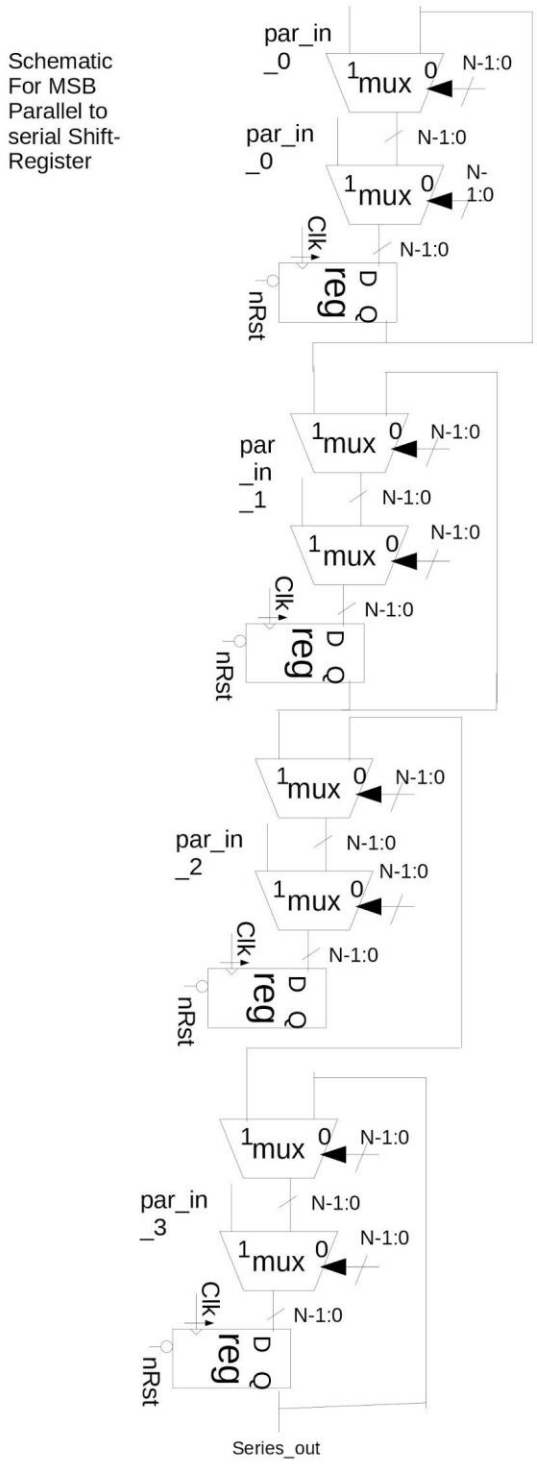

Last bit transmitted
8
Data_in
Shift Register
NRZI Encoded Data
8
Clk
Rst

Figure 23: Parallel to Serial RT



4. Budget Analysis

Complete area analysis:

Core Area Calculations				
Name of Block	**Category**	**Gate/ FF Count**	**Area (um2)**	**Comments**
USB to Ethernet FIFO count regs	Reg. w/ Reset	30	72,000	3 10-bit counters
Ethernet to USB FIFO count regs	Reg. w/ Reset	33	79,200	3 counters
On-chip SRAM	On-chip SRAM	16096	1,207,200	Assumes 2012 B memory
FIFO status	Combinational	23	17,250	a 10-bit and a 11-bit comparator
USB Controller next state	Combinational	9	6,750	
NRZI encoder	combinational	8	6,000	
NRZI decoder	Reg. w/ Reset	6	14,400	
bit stuffer	combinational	4	3,000	
bit unstuffer	Reg. w/ Reset	4	3,000	
USB transmitter timer	Reg. w/ Reset	4	3,000	
USB receiver timer	Reg. w/ Reset	4	9,600	
parallel to serial shift register	Reg. w/o Reset	3	7,200	
series to parallel shift register	Reg. w/o Reset	2	2,700	
16-bit CRC checker	Reg. w/	3	7,200	

	Reset			
5-bit CRC checker	Reg. w/ Reset	3	7,200	
Transmitter controller next-state	combinatio nal	9	6,750	
Receiver controller next-state	combinatio nal	9	6,750	
Transmitter controller state reg	Reg. w/ Reset	4	9,600	
Receiver controller state reg.	Reg. w/ Reset	4	9,600	
USB State Register	Reg. w/ Reset	4	9,600	Assume 15 states
MAC Receiver 32-bit CRC checker	Reg. w/ Reset	3	7,200	
MAC Transmitter 32-bit CRC checker	Reg. w/reset	3	7,200	
MAC transmitter state reg	Reg w/reset	4	9,600	
MAC Receiver state reg	Reg w/reset	4	9,600	
MAC transmitter next state reg	combinatio nal	10	7,500	
MAC Receiver next state reg	combinatio nal	9	6,750	
Total Core Area			1,535,850	

Chip Area Calculations (units in um or um2)

Number of I/O Pads:	12		
I/O Pad Dimensions:	90	by	300
I/O Based Padframe Dimensions:	870	by	870
Core Dimensions	1,239	by	1,239
Core Based Padframe Dimensions:	2,139	by	2,139
Final Padframe Dimensions:	2,139	by	2,139

Final Chip Area:		

The overall goal of the design was to build a reliable Ethernet to USB bridge. With this goal in mind we decided to minimize the area and indirectly reduce the power consumption. The design's area can be split up into individual blocks. The USB device controller is a state machine for which for which four flip-flops were required. Additional area is required for the combinational logic to check the PID data type. The next block is the FIFO which takes up the most space in the design. It uses two SRAMs and six counters. This is exactly enough to receive one maximum payload from either side of the bridge. It also requires a 10-bit counter to keep track of the FIFO addresses and byte count. Two status blocks use this information to calculate the number of bytes left in the FIFO. The final block is the MAC. The state registers each have 4 flip-flops and the combinational logic, to check CRC, is estimated to have low area.

To further improve on the area goals there are few possibilities. The possibilities involve combining the receive and transmit blocks in one. However, this is architecturally infeasible. Our design is based on transmitting and receiving blocks acting independently. Our main decision to save area was to reduce the size of the FIFO as it occupies the most space. Overall our area estimates met our expectations and are appropriate to fit in a USB to Ethernet adapter.

Design timing analysis:

Starting Component	Propagation Delay (ns)	Combinational Logic	Propagation Delay (ns)	Ending Component	Setup Time or Propagation Delay (ns)	Total Path Delay (ns)	Target Clock Period (ns)
NRZI encoder input	0.1	Next - Value Logic	1.6	NRZI encoder output	0.1	1.8	2.5
Bit Stuff Input	0.1	Next - Value Logic	0.8	Bit Stuff output	0.1	1	2.5
USB transmitter Timer input	0.1	Next - Value Logic	0.8	USB transmitter output	0.2	1.1	2.5
Parallel to Serial Input	0.2	none	-	Parallel to Serial Output	0.2	0.4	2.5

Transmitter Controller next state Input	0.1	Next - Value Logic	2	Transmitter Controller next state output	0.1	2.2	2.5
Transmitter Controller Reg.Input	0.2	None	-	Transmitter Controller Reg. output	2	2.2	2.5
Receiver Input (D+, D-)	0.1	Reciever contoller next state logic	1.6	Serial to Parallel Reg.	0.6	2.3	2.5
SRAM FIFO	5	none	0	Parallel to Serial Output	0.2	5.2	2.5

The timing analysis did not reveal any particular concerning critical path as of the preliminary design. The current estimates are based on projected combinational delays that may change at the time of synthesis at which point the timing analysis has to be reviewed. Only one path was found in violation of the target 2.5 ns clock and that is the SRAM data being read or written by the USB Controller or the Ethernet MAC. Since the standards require for a specific data transmission speed, the clock must remain at 2.5 ns or a similar value, therefore the SRAM delay can be accounted for and solved by stalling the next processes until the data is read successfully.

The data paths through the USB receiver and transmitter did not raise concern like that path through the SRAM, but were also of notable length. This can be contributed to the signals that have to be processed while the data is being decoded/encoded and bit-unstuffed/bit-stuffed. To optimize these paths, it can be reviewed which blocks are processing the signals and if a more direct path can be taken to assert the signals to the blocks that require them.

5. Project Timeline and Division of Tasks

Alex will be responsible for monitoring progress and negotiating task distribution.

Week of 11/13
"Rough drafts" of modules will be written as assigned below:
USB receiver - Alex
USB transceiver - Akshay
USB controller & FIFO - Alejandro
MAC - Vishnu

Week of 11/20
Test benches will be written for respective modules and modules tested as assigned below:
USB receiver - Alex
USB transceiver - Akshay
USB controller & FIFO - Alejandro
MAC - Vishnu

Week of 11/27
Testing and optimization of modules will continue as well as the overall test bench will be written to test the full design. Each member is responsible for their assigned module as well as contributing to the overall design testbench. Preparation for final presentation will begin, each member is responsible for the documentation of their functional block and the other sections will be split up between members.

Week of 12/4
Presentation week, testing of design will commence. Final presentation will be finished up and reviewed.

6. Success Criteria

The following list of success criteria will be used to determine whether we have met our project objectives. There are two types of criteria being defined - fixed and project dependant. The fixed criteria are required of all projects, while the project dependant criteria have been specified by us for our USB 2.0 to Ethernet design.

6.1 Fixed Criteria: (Total of 12 SC Points)

1. (2 points) Test benches exist for transceiver and controller blocks as well as the overall design.
2. (4 points) Entire design synthesizes completely, without any inferred latches, timing arcs, and, sensitivity list warnings.
3. (2 points) Source and mapped version of the complete design behave the same for all test cases. The mapped version simulates without timing errors except at time zero.
4. (2 points) A complete IC layout is produced that passes all geometry and connectivity checks.
5. (2 points) The entire design complies with reasonable area, pin count, and clock rate targets.

6.2 Design Specific Success Criteria: (Total of 8 SC Points)

6. (2 points) Successful receive of a data packet from the host to the USB receiver and stored correctly in the FIFO.
7. (2 points) Successful transfer of data in the FIFO to MAC and transfer from the MAC to the PHY.
8. (2 points) Successful receive of a data packet from the Ethernet PHY to the MAC and correctly stored in the FIFO.
9. (2 points) Successful transfer of data in the FIFO to the USB transmitter and transmitted outbound to the host.

.