University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Sciences

EECS 150 Fall 2002

Project Checkpoint #1
Receiving Data Over Ethernet

1 Objective

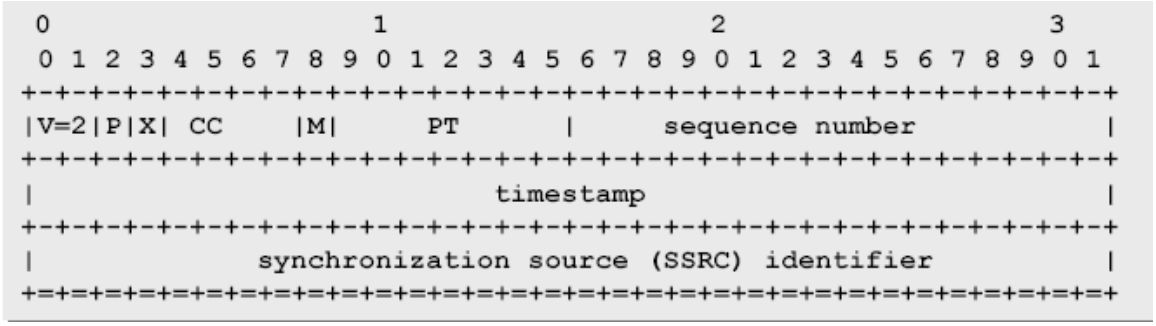
For this checkpoint, one will design and implement the components necessary for the processing of input data received in RTP transmitted over Ethernet. Support must be provided for selecting which packets one desires to receive data from, and for ignoring malformed or undesired packets. For the purposes of verification, every 64-bit word of data in an RTP packet will be accumulated and the resulting sum displayed on the Calinx board's 7 segment displays.

2 Specifications

Input comes into the FPGA from the Quad Ethernet Phy chip. For exact specifications on the Ethernet Phy, browse the assigned reading sections of the provided datasheet.

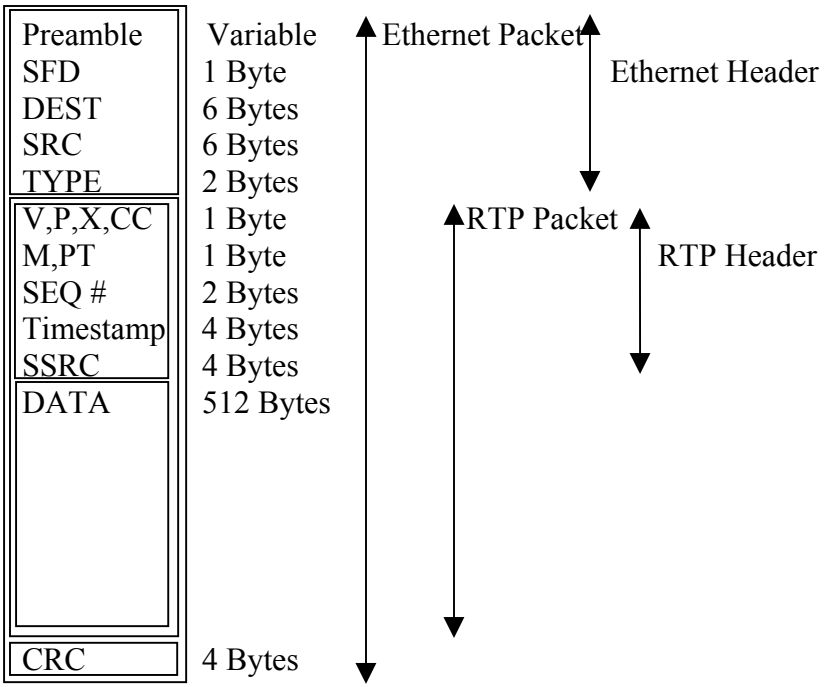
At top level, the data that one's design will be handling as input is an Ethernet packet. An Ethernet packet consists of a header, a data section, and a CRC field. The header consists of a variable number of nibbles of preamble (0101_2) followed by an SFD (1101_2), followed by 12 nibbles of destination address, 12 nibbles of source address, and 4 nibbles of type field, and the CRC field at the end of the Ethernet packet is 8 nibbles long. The data section in packets relevant to this checkpoint will be in the form of an RTP packet, and will be denoted as such by the type field of the Ethernet packet being 0x0107, with the destination field being all ones and the source field being unimportant.

The RTP packet consists of a 12 byte header, and, for those packets of the type one is interested in for this checkpoint, 512 bytes of data in addition to that header. The fields in the header are as follows:



RTP stands for real-time protocol, and is used in the transmission of real-time data streams. The **V**(ersion) field will always be 2 for all RTP packets. The **P**, **X**, **CC**, and **M** fields will be 0 for the purposes of this checkpoint. **PT**(payload type) for checkpoint 1 packets is 42_{10} = 0x2A = 010 1010_2. **Sequence number** is a monotonically increasing value that can be used to detect packets received out of order, but will not be used as such for this checkpoint. Similarly, the **timestamp** field is used for synchronizing with a real-time source, but its contents will be ignored for this checkpoint. The **SSRC** field contains an identifier for the source of the content provider. Valid **SSRC** values for this checkpoint range from 0 to 255.

Thus, a valid packet for checkpoint one would have the following form:

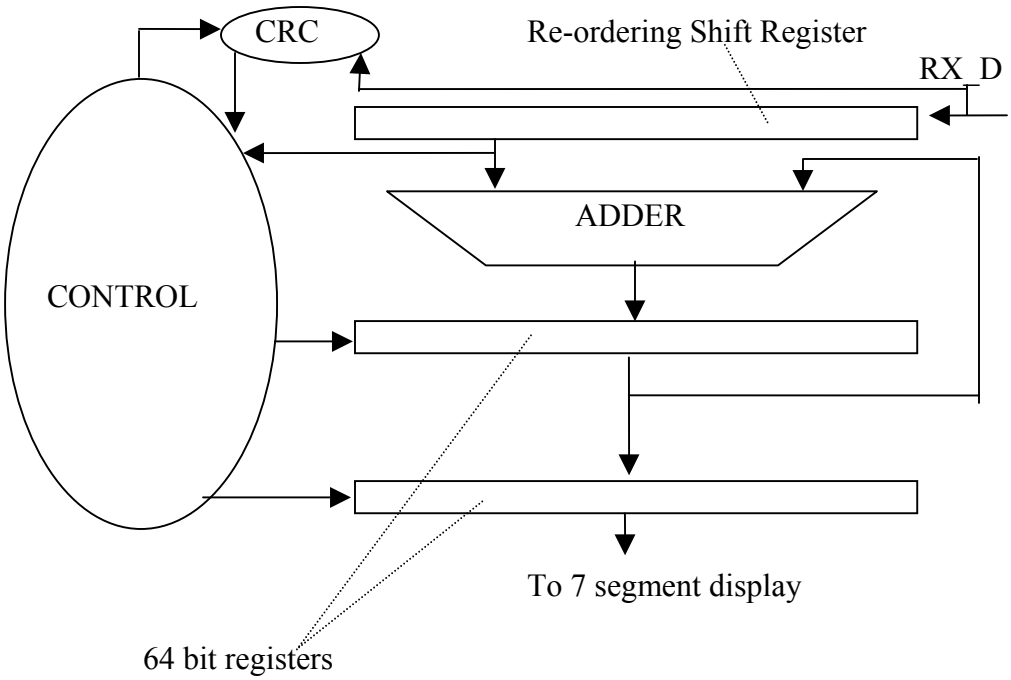


And would have

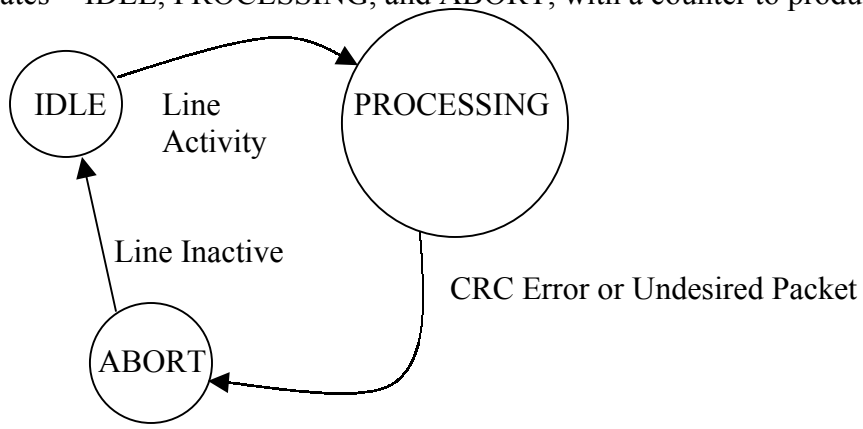
```
Preamble =    0x5
SFD =         0xD
DEST =        0xFFFFFFFFFFFF
SRC =         0x???????????
TYPE =        0x0107
VPXCC =       0x80
MPT =         0x2A
SEQ # =       0x????
Timestamp =   0x????????
SSRC =        0x000000??
```

3 Design Overview

The verilog files included in the zip file for this checkpoint define most of the pieces of the datapath one will use for this checkpoint. What remains for one to do is to connect these components appropriately and to construct the control logic that governs their behavior. When all is said and done, the design should bear some resemblance to the following:



For the control module, it is recommended that one use the following schema for one's FSM: 3 states - IDLE, PROCESSING, and ABORT, with a counter to produce substates.



4 Noteworthy Details

- The input to the CRC module has its bits flipped relative to the order they are coming in. (RX_D 3 -> CRC IN 0, 2 -> 1, etc)

- The ordering of the 2 nibbles within a byte is reversed. This issue is taken care of by a reordering done in the shift register. However, this means that the values in the shift register are only meaningful on byte aligned boundaries. On non-byte aligned nibble counts, the value of the first byte in the shift register will not be valid. This should not affect operation of the circuit at higher level, as major state transitions are byte aligned, but must be noted, as an error in the control of the shift register re-ordering would cause many problems.

- The CRC check module should be fed all nibbles of the packet except for those in the preamble or in the SFD. If the CRC check is successful, the signal CRCError will go low.

- A bit file has been provided that displays data being received on stream 0. One may use this to see what the stream 0 feed should look like when displayed.

5 Pre-lab

Draw out block diagrams of the interaction between the control logic and the datapath.

Work on testbench to simulate packet reception.

Read and understand the 6 verilog files provided in the zip file on the web.

6 Checkpoint

Download the zip file containing the six verilog files needed for this checkpoint. The file one will need to edit is receiver.v. However, one is responsible for understanding enough of what is going on in the other five files to be able to work with the sixth.

Before one plugs RJ45 connectors into one's Calinx board and attempts to debug one's design in hardware, one should make efforts to ensure that the sub-components of the design are functional. To this end, one should construct a testbench that walks the datapath through the processing of both valid and invalid Ethernet packets. When the datapath and the control block governing it appear to work in simulation, proceed to testing on the board.

For the purpose of testing one's design on the Calinx board, several streams of test data have been provided, with the stream identifiers numbered between 0 and 255. Each subnet carries one or more streams, with the stream numbers being assigned modulo the number of subnets (i.e. streams 1 and 5 are on subnet 1, there being 4 subnets). Here is a partial listing of test streams that may be particularly useful in debugging:

Stream 0: Data displayed should be the message "this is a test. this is only a test." scrolling to the left. This is the only stream being sent on subnet 0.

Stream 3: Data displayed should be the message: "flashing" and should flash at constant speed.

Stream 5: Data displayed should be the message " static " and should not move or otherwise change in any way.

The functionality of one's design for this checkpoint relies on both packet filtering and packet processing. While it is difficult to see the functioning of one without the functioning of the other, it is far from impossible. Stream 0 is the only data stream on subnet 0 and has been configured to contain far fewer garbage packets than any other of the other streams. Thus, even if one's packet filtering is not fully functional, Stream 0 can be used to debug one's packet filtering, assuming that packet processing is functional. Likewise, stream 5 contains much less dynamic data than any other stream, and so can be used for debugging packet processing, assuming that one's packet filtering is functional. Thus, what must be demonstrated for this checkpoint breaks down as follows:

Datapath:

- Demonstrate functionality of datapath components through simulation

Control logic:

- Demonstrate packet filtering on at least one stream
- Demonstrate packet processing on at least one stream

Robustness:

- Demonstrate functionality on all similar streams
- Demonstrate functionality on a stream with high garbage to data ratio

7 Acknowledgments

John W. Norm Z. Jack S. Fall 2002

Name: _____ Name: _____ Lab Section: ___

8 Check-offs

Verilog Design and Simulation

- Testbench and Simulation of packet reception _____(20%)_____(10%)

Onboard Demonstrations of:

- Packet Filtering _____(40%)_____(20%)
- Packet Processing _____(20%)_____(10%)
- Working with arbitrary streams _____(10%)_____(10%)
- Working with high garbage streams _____(10%)_____(5%)

Total TA: _____/_____ (%)