A quick guide to things that may not be entirely apparent. (Note: this is not a replacement for the writeup, nor does having read this mean you have sufficient knowledge to complete the checkpoint. I just happen to think this is useful data)

- Step one: remember what the goals of the checkpoint are:
  Receive packets and display the summation of the 64 bit segments of their RTP payloads on the LEDS

  Possible hazards to this occurring that must be overcome:

A garbled packet comes in	Solution: CRC check fails, ignore data
A packet from a stream you aren't displaying comes in	Solution: SSRC field is not as expected, ignore data
A packet of some other ether type (an ARP for instance) arrives	Solution: MAC type is not as specified ignore data
An RTP packet comes in with a payload type other than the one for checkpoint 1	Solution: RTP payload type is not as specified, ignore data

- 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.

- For quick reference:

Ethernet MAC type for eTV:	0x0107
Total RTP header length:	12 bytes = 24 nibbles
Checkpoint 1 RTP payload type:	0x2A
RTP payload length:	512 bytes = 1024 nibbles

- First things to think about when approaching the design of checkpoint 1:

  Draw out a block diagram of the circuit you hope to produce. Until you have an idea of what it is you want to create there is no point in attempting to use verilog to describe it. You will likely find it useful to make it clear which parts of your design manipulate data and which control the pieces that manipulate data and group them accordingly.