```
Test Rst for Router Core:

      A) Set RST_N = 1'b0 to reset the circuit.
      B) TX_Data_Valid should be 1'b0.
      C) RX_Data_Ready should be 1'b0.
      D) Packet_To_Node_Valid should be 1'b0.
      E) Core_Load_Ack should be 1'b0.
      F) If Router Address is 4'b0000 then core should create "The Token" and
begin circulating it.

RX Handshake Protocol:


      A)Set rst_n = 1'b0 to reset the circuit.
      B)Set rst_n = 1'b1 to resume.  Expect RX_Data_Ready to be 1'b0.
      C) Wait 50 cycles and make sure RX_Data_Ready does not go to 1'b1 until
RX_Data_Valid is 1'b0.
      D) Raise RX_Data_Valid to 1'b1.
      E) RX_Data_Ready should go to 1'b0 after a few cycles.
      F) Lower RX_Data_Valid to 1'b0.
            a)Lower after 300 cycles to ensure RX_Data_Ready does not go high until
RX_Data_Valid to 1'b0.
            b)Lower after 5 cycles to ensure Router Core can handle a fast
handshake.
      G)Repeat RX Handshake test with Router core clock running faster than TX/RX
Clock.
      H)Repeat RX Handshake test with Router core clock running slower than TX/RX
Clock.

TX Handshake Protocol:

      A)Set rst_n = 1'b0 to reset the circuit.
      B)Set rst_n = 1'b1 to resume. Expect TX_Data_Valid to be 1'b0.
      C)Wait 50 cycles to make sure TX_Data_Valid does not go to 1'b1 before
TX_Data_Ready goes to 1'b1.
      D)Force core into a state where it is ready to transmit data. Put 55'd 20 on
TX_Data vector.
      E)Expect TX_Data_Valid to go to 1'b1 shortly after Ready signal is raised.
      F)Lower TX_Data_Ready to 1'b0.
            a)Lower after 300 cycles to test a slow handshake.
            b)Lower after 5 cycles to test a fast handshake.
      G)Expect TX_Data_Valid to lower to 1'b0 shortly after.
      H)Raise TX_Data_Ready to 1'b1.
      I)With no valid data TX_Data_Valid should not go to 1'b1.
      J)Repeat TX Handshake test with Router core clock running faster than TX/RX
Clock.
      L)Repeat TX Handshake test with Router core clock running slower than TX/RX
Clock.

Router Core/Processor Node Protocol:

      Sending to Processing Node:
            A)Force Core into state where it is ready to transmit data.
            B)Packet_To_Node_Valid should go to 1'b1.
            C)Ensure that Packet_To_Node_Valid lowers after 1 cycle.

      Recieving from Processing Node:
            A)Place 29'd 20 on Packet_From_Node vector.
            B)Raise Packet_From_Node_Valid to 1'b1.
            C) 2 Cases: When the router core is ready for data and when it is not
ready.
                  a) If core is ready for data expect Core_Load_Ack to raise to
1'b1 for only 1 clock cycle.
                  b) If core is NOT ready for data expect Core_Load_Ack to stay
1'b0 until the core is ready.

Sub Module Encoder: //Tests that the encoder works properly

      A) Give the encoder 25 bits of unencoded payload.
      B) First set the 25th bit to 0 and make sure it is encoded with Checksum
properly.
      C) Retest  encoder and set 25th bit to 1 and make sure it is encoded with 3
of 6 properly.
      *Note: Will work out encoding by hand to make sure it works correctly.

Sub Module Decoder: //Tests that the decoder works properly

      A) Once encoder is tested encode 24 bits of payload with both Checksum and 3
of 6 algorithms.
      B) Give the Checksum encoded payload to decoder with 25th bit 1'b0 and make
sure it decodes correctly.
      C) Give the 3 of 6 encoded payload to decoder with 25th bit 1'b1 and make
sure it decodes correclty.
      *Note: Check with decoding by hand.
      D) Give both decoders incorreclty encoded data. Make sure the decoders
produce error signal of 1'b1 and sends a NACK.

Asynchronous Reset Tests:
      //These tests make sure resetting works at any point in the system
      A)Test Reset during Core to TX Handshake
      B)Test Reset during Core to RX Handshake
      C)Test Reset during encoding
      D)Test Reset during decoding
      E)Test Reset during data transfer from processing node to core
      F)Test Reset during data transfer from core to processing node

Router Core Packet Distribution:

      I) Receiving an ACK packet from RX.
            a) If we have the token. Expecting ACK after transmitting data.
                  i) Buffer lock on earliest packet sent should be released.  (i.e.
the packet
                   was correctly delivered.)
            b)If we dont have the token
                  i) We expect to see TX_Data_Valid to go to 1'b1, and TX_Data
should be an ACK packet.

      II) Receiving a TOKEN packet from RX.
            a) Expect the internal state to change, reflecting that the router core
is now ready to transmit data.
                  i.) If Packet_From_Node_Valid is high (processing node is waiting
to send data)
                        1.)  Expect TX_Data_Valid to go high and TX_Data to hold a
prepared packet.
                  ii.) If Packet_From_Node_Valid is low (processing node is *not*
trying to send data)
```

1.) Expect TX_Data_Valid to go high and TX_Data to hold a TOKEN packet.

III) Receiving a NACK packet from RX.

a) If we have the token, NACK means we need to resend the buffer.

i) Expect TX_Data_Valid to be high, and TX_Data will be the packet that needs to be resent.

b) If we do not have the token.

i) Expect TX_Data_Valid to be high, and TX_Data will be a NACK packet.

IV) Receiving data packet. (Do this for both types of encoding).

a) If we have the token, receiving a data packet indicates a breach of token-ring protocol, or
we provided a packet with a bad address. In either case, we expect to see the buffer unlock this data.

b) If do not have the token:

i) If the address matches the router core's address

1.) We expect to see Packet_To_Node_Valid go high, and decoded packet on Packet_To_Node[23:0]

ii) If the address does not match the router core's address

1.) We expect to see TX_Data_Valid go high once TX_Data_Ready is high, and TX_Data will be the encoded packet.

iii) If the address matches the router core, but decoded data does not pass checksum

1.) We expect to see TX_Data_Valid go high once TX_Data_Ready is high, and TX_Data will be NACK.