Sequence_number	Output	This is 16-bit signal, it represents the extracted sequence number of the packet.
Output_port	output	This is 5-bit signal, this represents the port from where packet is received in order to send an acknowledgement.
Send_wr_ack	output	This is 1-bit signal, Oncewrite_complete is asserted this logic should assert send_wr_ack signal.
Ack_sent	Input	This is 1-bit signal, Oncesend_wr_ack is asserted is driven by the remote interface to deassert the send_wr_ack from the logic.
clk	Input	This is a 1-bit clock signal for the logic.
rst	Input	This is a 1-bit reset signal for the logic.

Design description:
In this design, you need to extract the relevant data from the incoming packet based on the table number. You also need to assert the send_wr_acksignal once the data is extracted and written in the respective registers. Write packet format is shown in Fig. 13.

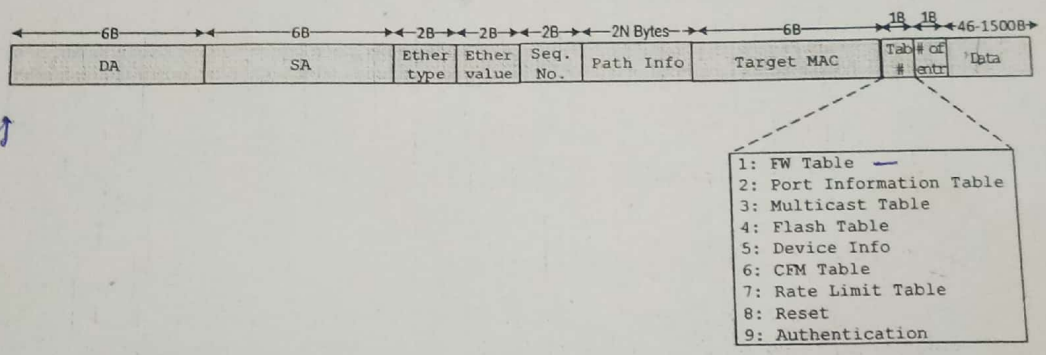

Fig. 13. Write Frame format

Tasks to perform:

1. Check the idata_av signal for logic high.
2. Check if wr_complete signal is logic high.
3. Check if opcode input to logic is = 3.
    a. In case opcode=0x"3" start reading the packet by asserting the iRd_Data='1';
4. Extract the frame sequence number.
5. Extract the table number and table address information from the packet.
6. Identify the data present in the packet.
7. Based on the table number extract the relevant information from the data present in packet.
8. Give the extracted data, table no and table address as output.
9. Once you get wr_complete='1', assert send_ACK signal and give frame sequence numberand output_portas output of your logic.
10. Identify the end of the packet marker. Once you find the end of packet marker assert "Rd_opcode" signal for 1 clock cycle.
11. Wait for wr_completeandAck_sentto be asserted.
12. Create the FSM based on above tasks.

Problem statement 6:

Design write logic in VHDL using Xilinx ISE. Write a testbench that include all possible test cases in order to verify your design using simulation. Implement your verified design on the ATLYS board to verify the proper functionality of your design on hardware.

Top-Level block diagram for your design is shown in Fig.12. The descriptions of the interfaces are given in table-VII. For design of your logic you need to use only the given interfaces at the top level of the design. Internally you can have temporary signals of your choice.

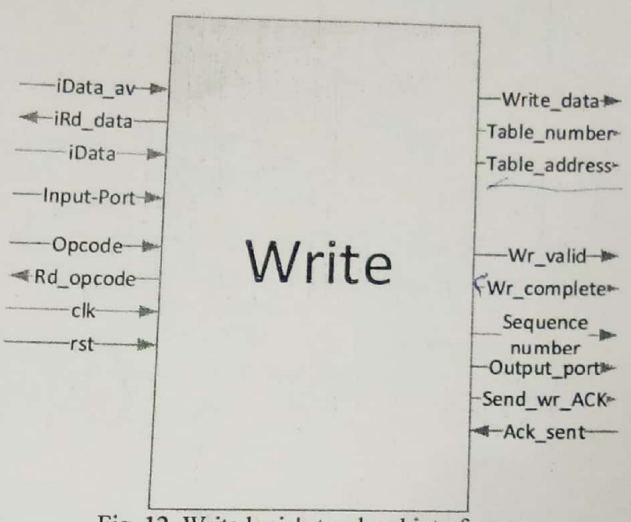

Fig. 12. Write logic's top-level interfaces.

Table VII. signal Description.

Interface	Direction	Description
iData_av	Input	This is a N+2-bits signal. A logic high represents the availability of the data for this module at the remote interface.
iRd_Data	Output	This is a N+2-bits signal. This signal is driven by your logic for reading data from remote interface. At a time only one of the N+2-bits can be high.
iData	Input	This is a (N+2)x144 bit signal. This is the incoming data to your logic. You receive a valid data for each clock cycle when you assert respective iRd_Data signal in your logic.
opcode	Input	This is a 3-bits signal, it represents the type of packet which is available at the iData_av.
Rd_opcode	output	This is 1-bit signal. your logic drive this signal high in case the opcode=3.
Wr_valid	Output	This is a 1-bit signal. A logic high represents the availability of the extracted data at the output interface of your logic. This signal is driven by your logic.
Wr_complete	Input	This is a 1-bit signal. This signal is driven high by remote logic once the extracted data is written in respective registers.
Write_Data	Output	This is a 512 bit signal. This is the outgoing extracted data aligned to LSB from your logic. Extracted data is valid when the wr_valid is at logic_high.
Input_port	Input	This is a 5-bit signal. this represents the port from where packet is coming to logic.
Table_number	Output	This is an 8-bit signal. This represents the table number where registers need to be written.
Table_address	Output	This is a 16-bits signal. This represents the offset location in table to write the registers