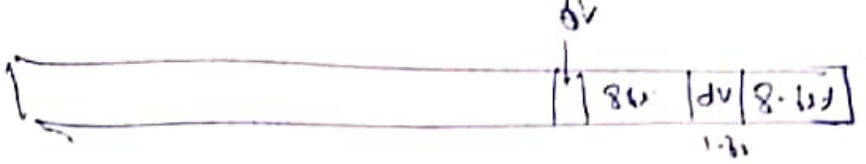

Problem statement 8:

Design frame_generate and forward 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.16. The descriptions of the interfaces are given in table-IX. 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.

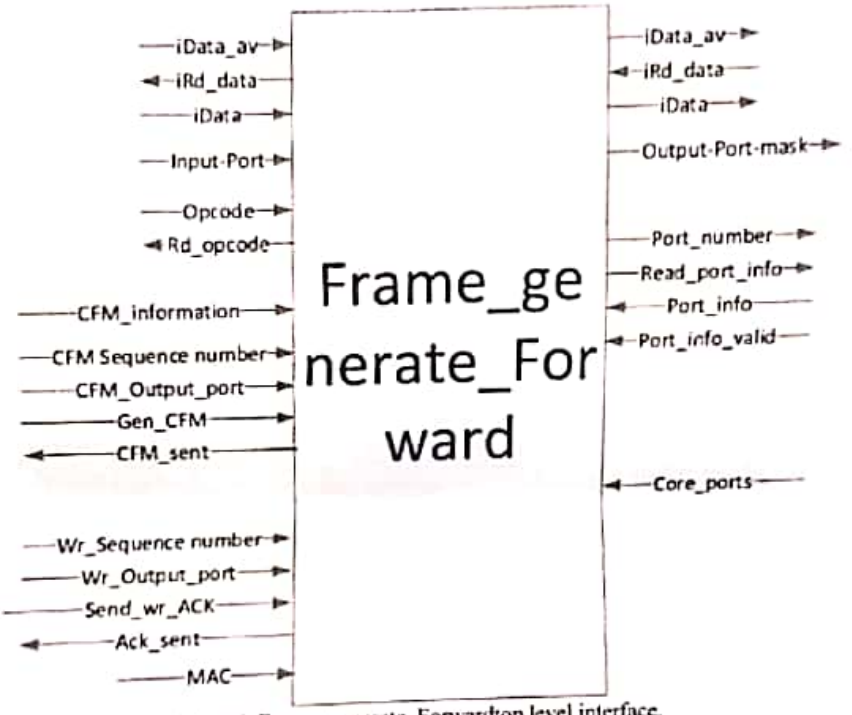

Fig. 16. Frame_generate_Forwardtop level interface.

Table IX. Signal Description

Interface	Direction	Description
iData_av	Input	This is a1-bit signal. A logic high represents the availability of the data for this module at the remote interface.
iRd_Data	Output	This is a -bit signal. This signal is driven by your logic for reading data from remote interface.
iData	Input	This is a 144 bit signal. This is the incoming data to your logic. You receive a valid data for each clock cycle when you assert iRd_Data signal in your logic.
oData_av	Output	This is a 1-bit signal. A logic high represents the availability of the data at the output interface of your logic. This signal is driven by your logic.
oData_rd	Input	This is a 1-bit signal. This signal is driven by remote logic for reading data from your logic.
oData	Output	This is a 144 bit signal. This is the outgoing data from your logic. You provide a valid data for each clock cycle when you assert oData_rdsignal is high
Output_port_mask	Output	This is a 32-bit signal. Each bit represents an output port. If a bit

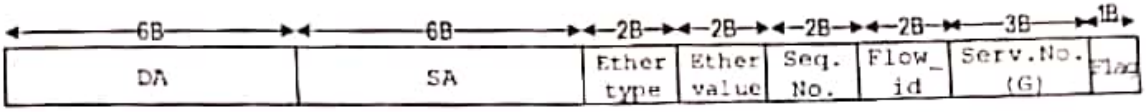

Fig. 17. CFM Frame format

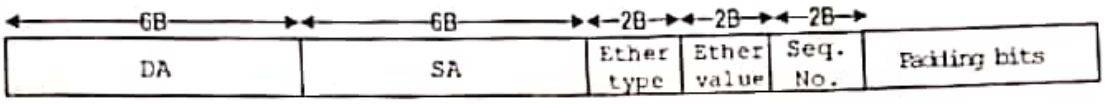

Fig. 1812. write ack message format.

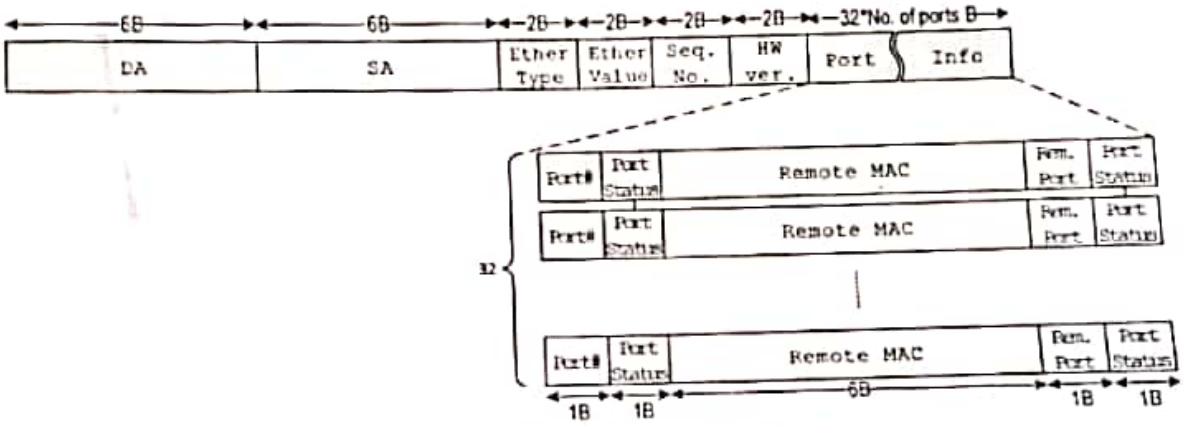

Fig. 139. Hello ack message format

Tasks to perform:

1. Check the `idata_av` signal for logic high.
2. Check if `FSM_bsy`signal of your design is low.
3. Check if `opcode` input to logic is = 1.
   a. In case `opcode=0x"1"` start reading the packet by asserting the `iRd_Data` ='1';
   b. Mark the FSM_bsy='1';
   c. Extract the sequence number from the packet.
   d. Forward this packet to all the `core ports` except the input port.
   e. Generate a `hello ack` frame using the extracted sequence number.
   f. Send this generated reply frame on the `output_port` by providing `output_port_mask(input_port)='1'`.
   g. Identify the end of the packet marker. Once you find the end of packet marker assert "`Rd_opcode`" signal for 1 clock cycle.
   h. Mark FSM_bsy='0';
4. In case no data is available on `idata_av`.
5. Check if `send_wr_ack` or `gen_cfm` signals are high.
   a. If the signals are high, Mark the FSM_bsy='1';
   b. Use the respective `sequence number` and other relevant information to generate the message.
   c. Provide the output port by masking respective bit in `output_port_mask` signal to '1'
   d. Mark `FSM_bsy`='0';
6. Create the FSM based on above tasks.

		is asserted high, available data should be sent on the respective output port.
MAC	Input	This is a 48-bit signal. This represents the MAC address of your design.
Opcode	Input	This is a 3-bits signal. This represents classification of incoming packet.
Rd_opcode	Output	This is a 1-bit signal to read the opcode from your logic.
Input_port	Input	This is a 5-bit signal representing the input port of the packet.
Port_mask	Input	This is a 32-bit signal, where index of each bit represents an output port. Respective index bit is made high by the remote logic representing the destination port for the packet. otherwise it remains at logic low.
Port_number	Output	This is 5-bit signal, indicating the port number for reading the port's status.
Rd_port_info	Output	This is 1-bit signal. This is asserted to request a port's status where port number is available on Port_number signal.
Port_info	Input	This is an 80-bit signal. This provides status of the requested port.
Port_info_valid	Input	This is a 1-bit signal, it represents that the status available on Port_info signal is valid.
CFM_Sequence_number	Input	This is 16-bit signal, it represents the sequence number for the CFM packet to be generated.
Output_port	Input	This is 5-bit signal, this represents the port where generated CFM m to be sent.
CFM_Information	Input	This is a 256-bit, information. This information is used to generate the CFM frame.
Gen_CFM	Input	This is 1-bit signal, input to this logic. This signal represents a request to generate the CFM frame.
CFM_sent	Output	This is 1-bit signal, it is sent in response to Gen_CFM. Once the CFM frame is generated and sent to respective output port.
Wr_Sequence_number	Input	This is 16-bit signal, it represents the sequence number of the wr ack packet.
Output_port	Input	This is 5-bit signal, this represents the port, where generated wr_ack packet need to be sent.
Send_wr_ack	output	This is 1-bit signal, This signal is request to this logic to generate the Wr_ack message.
Ack_sent	Input	This is 1-bit signal, In response to Wr_ack message this signal is asserted by this logic to indicate that ack message is sent..
Core_ports	Input	This is a 32 bit signal. If a bit is high in this signal, it represents that the respective port is marked as core port.
MAC	Input	This is 32-bit signal. It is the MAC address of the design.
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 generate various messages and send them to desired port. This logic is also used for forwarding an incomingpacket to desired output port. Output port is provided by masking the index of output port in Output_port_mask signal. Generated messages are stored in a FIFO. Remote logic read the message from output fifo based on the availability of the data in fifo. Format of different messages to be generated are shown in Fig 17-19.