Case Study: FPGA-Based Ethernet Packet Parser Using AXI-Stream

Scenario

The objective of this project is to design an FPGA module that efficiently processes Ethernet frames received via an AXI-Stream interface. The module must extract essential fields from the Ethernet, IPv4, and UDP headers while ensuring minimal latency and supporting high-throughput operation. Extracted header information should be forwarded to a processing unit, while the payload is transmitted without modification.

System Requirements

Packet Reception

- Ethernet frames arrive via an AXI-Stream interface consisting of tdata, tvalid, tready, tlast, and tkeep signals.
- The frame sizes range from a minimum of 64 bytes to a maximum of 1500 bytes.
- The frame starts form Destination MAC Address and end with last payload.
- The FPGA module must support processing at line rates of at least 10GbE.

Packet Parsing

The FPGA module must extract the following fields:

Ethernet Header (14 bytes)

- Destination MAC Address (6 bytes)
- Source MAC Address (6 bytes)
- EtherType (2 bytes)

IPv4 Header (Minimum 20 bytes)

- Source IP Address (4 bytes)
- Destination IP Address (4 bytes)
- Protocol (1 byte)

UDP Header (8 bytes)

- Source Port (2 bytes)
- Destination Port (2 bytes)
- Length (2 bytes)

Payload Processing

- The payload should be forwarded to the next processing unit without any modifications.

Deliverables

- RTL Design
- Testbench

Module Interface Specification

Module: ethernet_parser

Inputs

- clk: Clock signal.
- rst_n: Active-low reset.

AXI-Stream Input (From MAC/Network)

- s_axis_tdata [63:0]: Input data bus.
- s_axis_tkeep [7:0]: Byte-valid indicator for input data.
- s_axis_tvalid: Valid signal for input data.
- s_axis_tready: Ready signal to accept input data.
- s_axis_tlast: End of frame indicator.

AXI-Stream Output (Parsed Data Forwarded)

- m_axis_tdata [63:0]: Output data bus.
- m_axis_tkeep [7:0]: Byte-valid indicator for output data.

- m_axis_tvalid: Valid signal for output data.
- m_axis_tready: Ready signal to accept output data.
- m_axis_tlast: End of frame indicator.

Parsed Header Outputs

- parsed_valid_pulse: One-cycle pulse indicating valid parsed data.
- dst_mac [47:0]: Extracted Destination MAC Address.
- src_mac [47:0]: Extracted Source MAC Address.
- ethertype [15:0]: Extracted EtherType field.
- src_ip [31:0]: Extracted Source IP Address.
- dst_ip [31:0]: Extracted Destination IP Address.
- src_port [15:0]: Extracted Source Port.
- dst_port [15:0]: Extracted Destination Port.

Implementation Considerations

- The parsed_valid_pulse output must be a single-cycle pulse indicating that valid parsed header information is available.
- The design should ensure minimal processing latency while maintaining data integrity.
- The module must handle out-of-order packet arrival and ensure that headers are correctly extracted.
- Proper synchronization must be maintained between data input and parsed output to prevent data loss or corruption.

This design will be tested using a structured testbench to validate functionality under various traffic conditions, including stress testing at 10GbE speeds.

Verilog Module Template

```
module ethernet_parser (
  input wire clk,
  input wire rst_n,

  // AXI-Stream Input (From MAC/Network)
  input  wire [63:0]  s_axis_tdata,
  input  wire [7:0]   s_axis_tkeep,
  input  wire         s_axis_tvalid,
  output wire         s_axis_tready,
  input  wire         s_axis_tlast,

  // AXI-Stream Output (Parsed Data Forwarded)
  output reg  [63:0]  m_axis_tdata,
  output reg  [7:0]   m_axis_tkeep,
  output reg          m_axis_tvalid,
  input  wire         m_axis_tready,
  output reg          m_axis_tlast,

  // Parsed Header Outputs
  output reg          parsed_valid_pulse,
  output reg  [47:0]  dst_mac,
  output reg  [47:0]  src_mac,
  output reg  [15:0]  ethertype,
  output reg  [31:0]  src_ip,
  output reg  [31:0]  dst_ip,
  output reg  [15:0]  src_port,
  output reg  [15:0]  dst_port
);

// Module implementation to be added here

endmodule
```