CE387: Real-Time Digital Systems Design and Verification with FPGAs
Professor David Zaretsky
Assignment 5
UDP Parser

Matias Ketema
tnc5178

Simulation Results

- **Clock cycle count:** 4320.5 cycles (got from Time(ns)/10ns)
- **Errors reported:** 0
- **Functional Coverage:** 81.25%

Synthesis Results

- **Maximum frequency**: The synthesis report estimates a maximum operating frequency of 142.2 MHz.
- **Registers /LUTs/Logic Elements**: The design utilizes a total of 166 registers on the device. It also consumes 470 combinational functions (Logic Elements/LUTs).
- **Memory utilization**: The design uses 40,960 memory bits (Total ESB).
- **Multipliers (DSPs)**: No Digital Signal Processing blocks implemented
- **Worst path (timing analysis)**: The worst path has a negative slack of -1.055 nanoseconds with a total propagation delay of 7.559 ns. This critical timing path begins at the memory output of the input control FIFO (fifo_in.ctrl_fifo.fifo_buf) and ends at the byte counter register in the parser instance (udp_parser_inst.byte_cnt[15]).
- **Schematic architecture (RTL)**: This is a streaming packet processing architecture. The design (udp_parser_top) wraps a central processing core (udp_parser_inst) with input and output buffering. The buffering is split into distinct control and data paths (evidenced by fifo_in.ctrl_fifo and fifo_in.data_fifo), allowing for the separation of packet metadata and payload. The parser utilizes a Finite State Machine (FSM) with at least 6 states (encoded as one-hot) to manage the parsing logic.

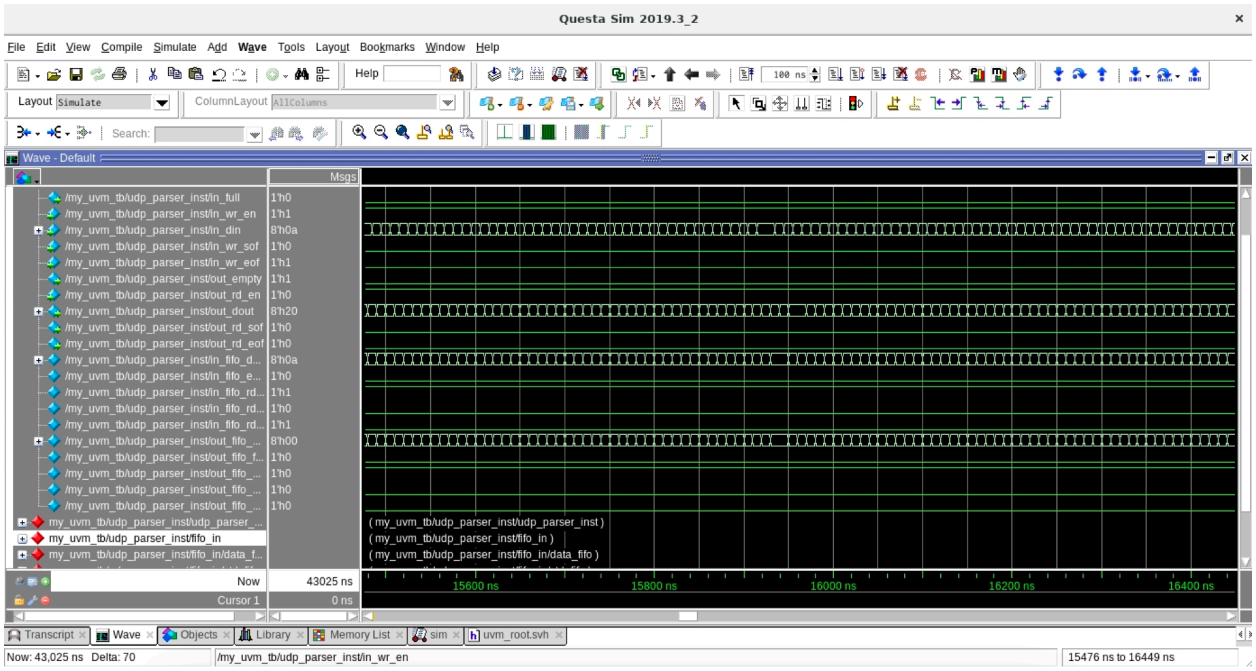
This project implements a UDP packet parser in SystemVerilog that strips Ethernet, IP, and UDP headers from raw packet data and outputs only the UDP payload. The design was heavily inspired by the provided grayscale reference project, reusing its FIFO infrastructure, UVM testbench structure, and simulation scripts. The system was verified using the Universal Verification Methodology to ensure that the extracted payload bytes are bit-true against a known-good reference generated directly from the input PCAP file.

The core of the design is the UDP parser module, which is implemented as a finite state machine with six states. The FSM reads raw packet bytes one at a time from the input FIFO and walks through each protocol layer sequentially. During the Ethernet header state, it consumes 14 bytes and captures the EtherType field to verify the packet is IPv4 by checking for 0x0800. If the check fails, it transitions to the FLUSH state which discards the rest of the packet until EOF. The IP header state parses 20 bytes and validates that the version is 4 and the protocol is 0x11 for UDP. The UDP header state parses 8 bytes and calculates the payload size from the length field. Finally, the UDP_DATA state forwards the exact number of payload bytes to the output FIFO, asserting SOF on the first byte and EOF on the last. The top-level module wraps this parser between an input and output fifo_ctrl, which are the same modules from the grayscale project, each containing two parallel FIFOs for data and SOF/EOF control signals.

The UVM testbench follows the same architecture as the grayscale reference. The sequence reads a PCAP file in binary mode and sends each packet through the driver with SOF and EOF markers. The output monitor reads parsed data from the output FIFO and writes every byte to an output.txt file. A compare monitor loads expected reference data, and the scoreboard does byte-by-byte comparison. The scoreboard also includes functional coverage using a covergroup that bins output data values into six ranges, tracks SOF and EOF assertion, and crosses them to check all four combinations. The test data consists of four UDP packets totaling 3979 bytes of payload, and coverage came out to 81.25% because no packet had a single-byte payload, leaving the SOF=1/EOF=1 cross bin empty.

The main challenge during verification was that the simulation initially reported 3009 errors out of 3979 bytes. Looking at the mismatch pattern showed the expected bytes contained sequences like 0xEF 0xBF 0xBD, which is the UTF-8 replacement character. The reference file test_output.txt had been generated by a C program that prints raw bytes using printf with %c, and at some point the output passed through a text pipeline that replaced every non-ASCII byte with a 3-byte UTF-8 sequence, shifting everything after it. Regenerating the reference file by extracting UDP payloads directly from the PCAP binary fixed the issue, and the simulation passed with zero errors.

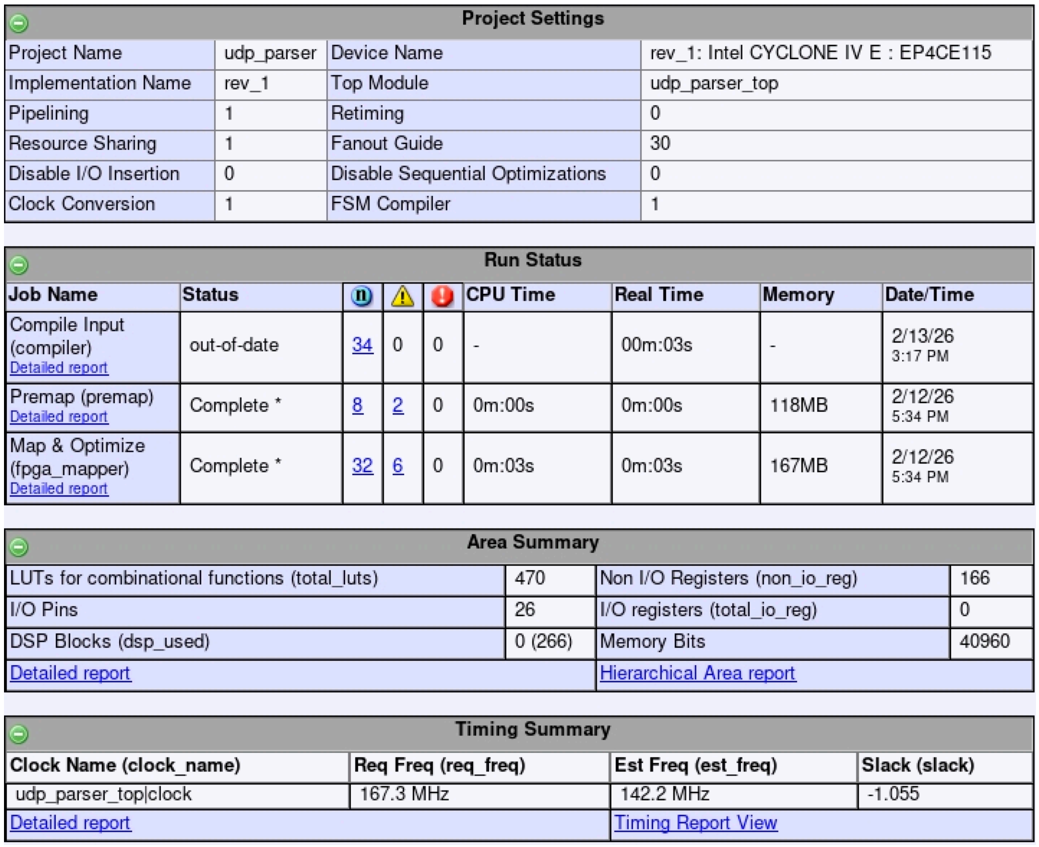
Simulation Proof:



	Msgs
/my_uvm_tb/udp_parser_inst/in_full	1'h0
/my_uvm_tb/udp_parser_inst/in_wr_en	1'h1
/my_uvm_tb/udp_parser_inst/in_din	8'h0a
/my_uvm_tb/udp_parser_inst/in_wr_sof	1'h0
/my_uvm_tb/udp_parser_inst/in_wr_eof	1'h1
/my_uvm_tb/udp_parser_inst/out_empty	1'h1
/my_uvm_tb/udp_parser_inst/out_rd_en	1'h0
/my_uvm_tb/udp_parser_inst/out_dout	8'h20
/my_uvm_tb/udp_parser_inst/out_rd_sof	1'h0
/my_uvm_tb/udp_parser_inst/out_rd_eof	1'h0
/my_uvm_tb/udp_parser_inst/in_fifo_d...	8'h0a
/my_uvm_tb/udp_parser_inst/in_fifo_e...	1'h0
/my_uvm_tb/udp_parser_inst/in_fifo_rd...	1'h1
/my_uvm_tb/udp_parser_inst/in_fifo_rd...	1'h0
/my_uvm_tb/udp_parser_inst/in_fifo_rd...	1'h1
/my_uvm_tb/udp_parser_inst/out_fifo_...	8'h00
/my_uvm_tb/udp_parser_inst/out_fifo_f...	1'h0
/my_uvm_tb/udp_parser_inst/out_fifo_...	1'h0
/my_uvm_tb/udp_parser_inst/out_fifo_...	1'h0
/my_uvm_tb/udp_parser_inst/out_fifo_...	1'h0
my_uvm_tb/udp_parser_inst/udp_parser_...	(my_uvm_tb/udp_parser_inst/udp_parser_inst)
my_uvm_tb/udp_parser_inst/fifo_in	(my_uvm_tb/udp_parser_inst/fifo_in)
my_uvm_tb/udp_parser_inst/fifo_in/data_f...	(my_uvm_tb/udp_parser_inst/fifo_in/data_fifo)
Now	43025 ns
Cursor 1	0 ns

Now: 43,025 ns Delta: 70 /my_uvm_tb/udp_parser_inst/in_wr_en 15476 ns to 16449 ns

Synthesis Screenshot



Project Settings

Project Name	udp_parser	Device Name	rev_1: Intel CYCLONE IV E : EP4CE115
Implementation Name	rev_1	Top Module	udp_parser_top
Pipelining	1	Retiming	0
Resource Sharing	1	Fanout Guide	30
Disable I/O Insertion	0	Disable Sequential Optimizations	0
Clock Conversion	1	FSM Compiler	1

Run Status

Job Name	Status	ⓘ	⚠	❗	CPU Time	Real Time	Memory	Date/Time
Compile Input (compiler) Detailed report	out-of-date	34	0	0	-	00m:03s	-	2/13/26 3:17 PM
Premap (premap) Detailed report	Complete *	8	2	0	0m:00s	0m:00s	118MB	2/12/26 5:34 PM
Map & Optimize (fpga_mapper) Detailed report	Complete *	32	6	0	0m:03s	0m:03s	167MB	2/12/26 5:34 PM

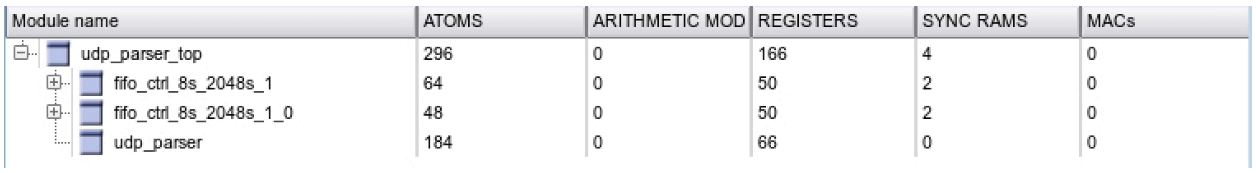
Area Summary

LUTs for combinational functions (total_luts)	470	Non I/O Registers (non_io_reg)	166
I/O Pins	26	I/O registers (total_io_reg)	0
DSP Blocks (dsp_used)	0 (266)	Memory Bits	40960
Detailed report		Hierarchical Area report	

Timing Summary

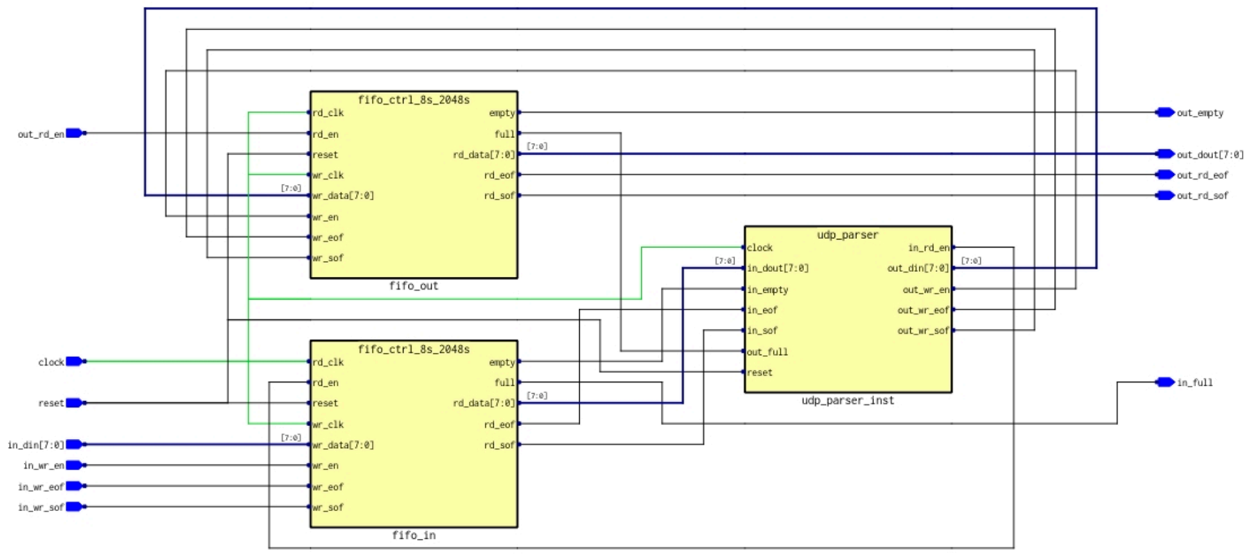
Clock Name (clock_name)	Req Freq (req_freq)	Est Freq (est_freq)	Slack (slack)
udp_parser_top|clock	167.3 MHz	142.2 MHz	-1.055
Detailed report		Timing Report View	

Hierarchical Area Report:

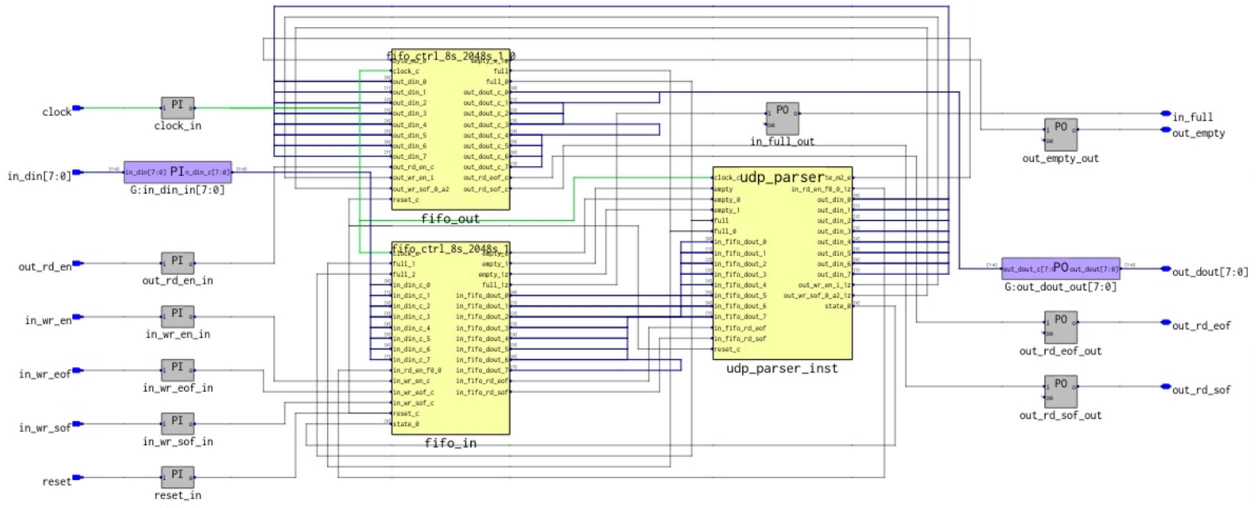


Module name	ATOMS	ARITHMETIC MOD	REGISTERS	SYNC RAMS	MACs
udp_parser_top	296	0	166	4	0
fifo_ctrl_8s_2048s_1	64	0	50	2	0
fifo_ctrl_8s_2048s_1_0	48	0	50	2	0
udp_parser	184	0	66	0	0

RTL Hierarchical:



Technology Hierarchical:



RTL Flattened: