DIGITAL SYSTEM DESIGN FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA

IPv4 Network Packet

Header Processor

GROUP PA-07

Kamila Salma Fathiyya 2406487071

David Alexander 2406351592

Nicholas Edmund Tanaka 2406352986

Tubagus Dafa Izza Fariz 2406350122

PREFACE

Kami bersyukur kepada Tuhan Yang Maha Kuasa atas rahmat dan penyertaan-Nya sehingga kami dapat menyelesaikan laporan akhir proyek ini yang berjudul "IPv4 Header Processing Accelerator". Laporan ini disusun sebagai bagian dari tugas akhir mata kuliah Perancangan Sistem Digital, dengan tujuan menerapkan berbagai konsep desain perangkat keras yang telah kami pelajari sepanjang praktikum.

Proyek ini dirancang untuk mensimulasikan sebuah modul pemrosesan header jaringan yang bekerja seperti akselerator pada perangkat router. Modul ini melakukan beberapa fungsi utama, yaitu memverifikasi checksum header IPv4, mengurangi nilai Time To Live (TTL), membentuk ulang checksum baru, serta melakukan pengecekan firewall sederhana berdasarkan daftar alamat IP yang diblokir. Seluruh fungsi ini diimplementasikan menggunakan bahasa VHDL dengan mengintegrasikan berbagai gaya pemrograman dari modul-modul praktikum, termasuk dataflow, behavioral, structural, looping construct, procedure–function, finite state machine, serta microprogramming.

Dalam penyelesaian proyek ini, kami ingin menyampaikan terima kasih kepada dosen pengampu yang telah memberikan teori dasar serta arah pembelajaran, serta para asisten laboratorium yang telah membimbing dan membantu kami selama proses praktikum. Bimbingan tersebut sangat berperan dalam membantu kami memahami setiap tahap perancangan dan menyelesaikan kendala teknis yang muncul selama pengembangan.

Kami berharap laporan ini dapat memberikan manfaat sebagai bahan pembelajaran dan referensi bagi pengembangan sistem digital yang lebih kompleks di masa mendatang. Kami juga menyadari bahwa laporan ini masih memiliki kekurangan, dan kami terbuka terhadap kritik serta saran yang membangun demi perbaikan ke depannya.

Depok, December 07, 2025

Group PK-07

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Keamanan dan efisiensi adalah dua aspek paling krusial dalam jaringan komputer. Saat sebuah paket data IPv4 melintasi jaringan, router tidak hanya sekadar meneruskan data, tetapi juga harus melakukan serangkaian validasi yang intensif. Router harus mengurangi nilai Time-To-Live (TTL) untuk mencegah looping, menghitung ulang checksum untuk menjaga integritas data, dan yang paling penting, melakukan penyaringan (filtering) untuk memblokir alamat IP yang berbahaya atau tidak diinginkan.

Secara konvensional, fungsi firewall dan pemrosesan paket ini sering dilakukan oleh software yang running di atas sistem operasi. Namun, pendekatan ini memiliki keterbatasan latency dan throughput, terutama ketika menghadapi serangan jaringan atau data traffic yang sangat padat. CPU harus memproses interupsi untuk setiap paket, yang memakan banyak siklus instruksi. Oleh karena itu, solusi hardware-based menggunakan FPGA menjadi pilihan karena kemampuannya memproses data secara paralel dan deterministik.

Proyek akhir ini mengintegrasikan seluruh materi praktikum modul 2 hingga 9 menjadi sebuah sistem utuh: **IPv4 Packet Processor with Exact-Match Firewall**. Berbeda dengan desain sederhana, sistem ini ada arsitektur berbasis Microprogrammed Control, di mana alur pemrosesan diatur oleh instruksi mikro dalam ROM, bukan sekadar FSM biasa. Proyek ini tidak hanya mendemonstrasikan kemampuan manipulasi bit (TTL dan Checksum), tetapi juga implementasi firewall statis berbasis ROM yang efisien dan synthesizable.

1.2 PROJECT DESCRIPTION

Proyek ini bertujuan merancang sebuah blok pemroses perangkat keras yang menerima header IPv4 (160-bit) dan memprosesnya melalui beberapa tahapan validasi secara sekuensial. Sistem ini dirancang agar synthesizable untuk target FPGA menggunakan toolchain standar seperti Intel Quartus Prime dan disimulasikan menggunakan ModelSim.

Fungsi utama dari sistem ini meliputi:

1. **TTL Processing:** Mengurangi nilai Time-To-Live sebesar 1. Jika hasil pengurangan menjadi 0, paket dianggap exipired dan sistem akan mengaktifkan sinyal drop.
2. **Checksum Recomputation:** Karena nilai TTL berubah, Header Checksum harus dihitung ulang. Kami menggunakan pendekatan Dataflow (penjumlahan paralel) untuk menghitung checksum baru menggunakan logika ones-complement.
3. **Exact-Match Firewall:** Sistem memeriksa alamat Source IP dan Destination IP paket terhadap daftar 10 alamat IP terlarang yang disimpan dalam ROM (Read-Only Memory). Jika ada kecocokan (match), paket akan langsung ditolak.
4. **Microprogrammed Control:** Alur kerja sistem tidak dikendalikan oleh hardwired FSM biasa, melainkan oleh Micro-controller yang membaca instruksi dari Microcode ROM. Ini memungkinkan fleksibilitas kontrol alur (seperti mengaktifkan unit TTL, lalu Checksum, lalu Firewall) secara terstruktur.

Keputusan akhir (Output Logic) menggabungkan hasil dari ketiga unit tersebut: paket akan di-drop jika TTL habis, checksum salah, atau IP terdaftar dalam blacklist firewall.

1.3 OBJECTIVES

Tujuan kami mengerjakan proyek akhir ini adalah:

1. Merancang arsitektur sistem digital yang memisahkan Datapath (TTL Unit, Checksum Unit, Firewall Checker) dan Control Unit (Micro-controller) secara modular.
2. Mengimplementasikan **Exact-Match Firewall** menggunakan memori ROM untuk menyimpan daftar blokir IP secara efisien tanpa menggunakan RAM eksternal.
3. Menerapkan teknik desain **Microprogramming** (Module 9) untuk mengatur sekuensial eksekusi hardware yang kompleks.
4. Memverifikasi kebenaran logika desain menggunakan testbench komprehensif yang mencakup skenario corner cases (seperti TTL=1, Checksum corrupt, dan IP yang diblokir).
5. Memastikan seluruh kode VHDL memenuhi kaidah synthesizable dan sesuai dengan standar VHDL-2008 (atau kompatibel dengan VHDL-93).

1.4 ROLES AND RESPONSIBILITIES

Roles	Responsibilities	Person
Kamila Salma Fathiyya	Datapath	- Membantu perumusan ide dan menyusun garis besar struktur kode - Menyiapkan helpers dan mengerjakan unit perhitungan aritmatika untuk TTL dan Checksum. - Menyusun bab Implementation, serta melakukan proofread laporan.
Nicholas Edmund	Control Unit	- Merancang sistem kontrol utama berbasis *Microcontroller* - Menyusun instruksi mikro pada ROM. - Menyusun Introduction laporan
Tubagus Dafa	Features	- Mengembangkan logika Firewall untuk mendeteksi IP terblokir - Melakukan pengujian unit khusus modul tersebut. - Menyusun Testing and Analysis laporan

David Alexander	System Integrator	- Mengusulkan ide utama proyek serta mengelola github - Menyatukan seluruh modul menjadi sistem utuh, melakukan simulasi akhir, mengorganisir struktur kode dan menyusun dokumentasi - Menyusun Conclusion laporan

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

1. Model Sim

Tools pertama yang digunakan untuk simulasi dan verifikasi pada proyek kali ini adalah ModelSim (Intel FPGA Starter Edition). ModelSim digunakan untuk mensimulasikan perilaku fungsional dari desain packet processor sebelum dilakukan sintesis. Beberapa kegunaan utama dari ModelSim pada proyek ini antara lain:

- Simulasi Logika Sekuensial dan FSM ModelSim memungkinkan simulasi sinkronisasi clock untuk memvalidasi transisi Finite State Machine (FSM) pada modul Firewall dan logika sekuensial pada unit TTL. Ini penting untuk memastikan bahwa perubahan status dari IDLE ke CHECKING hingga DONE berjalan sinkron dengan tepi naik clock.
- Pengujian Komponen Modular (Unit Testing) Dengan ModelSim, setiap modul seperti ttl_unit, checksum_unit, dan firewall_rom_checker dapat diuji secara terpisah menggunakan testbench khusus. Hal ini memastikan bahwa logika aritmatika ones-complement pada checksum dan komparator pada firewall berfungsi dengan benar sebelum diintegrasikan.
- Simulasi Eksekusi Microcode (System Integration) ModelSim digunakan untuk mensimulasikan eksekusi instruksi mikro yang tersimpan dalam ROM. Simulasi ini mencakup visualisasi gelombang (waveform) untuk memverifikasi alur pemrosesan paket IPv4 dari input hingga output, serta memvalidasi sinyal keputusan drop berdasarkan kondisi TTL, checksum, atau firewall.

2. Quartus

Tools kedua yang digunakan pada proyek kali ini yaitu Intel Quartus Prime. Quartus sendiri digunakan sebagai software utama untuk melakukan proses synthesis dan validasi desain RTL (Register Transfer Level) dari kode VHDL yang telah dirancang. Melalui Quartus, kode VHDL yang ditulis dapat diubah menjadi skema logika digital yang terstruktur dan dapat divisualisasikan dalam bentuk RTL Viewer.

Proses synthesis ini memungkinkan integrasi antara Unit Kontrol (micro_controller) dan Unit Datapath (ttl_unit, checksum_unit, firewall_rom_checker) menjadi sebuah top-level entity (packet_processor). Dengan visualisasi ini, interkoneksi jalur data 160-bit dan sinyal kontrol dapat dianalisis untuk memastikan desain memenuhi kaidah synthesizable dan bebas dari kesalahan logika.

3. **Visual Studio Code**

Visual Studio Code digunakan sebagai text editor utama untuk penulisan kode sumber VHDL dan dokumentasi proyek. Selain digunakan untuk menulis entitas dan arsitektur VHDL, VS Code juga digunakan untuk membuat file Markdown (README.md) yang berisi spesifikasi arsitektur, peta memori Microcode ROM, serta penjelasan cara kerja sistem. Penggunaan ekstensi pendukung pada VS Code membuat proses penulisan sintaks VHDL dan dokumentasi teknis menjadi lebih efisien dan terorganisir.

2.2 IMPLEMENTATION

2.2.1 Modul 2: Dataflow Description (Checksum Unit)

Modul ini berkaitan dengan perancangan logika kombinasional menggunakan gaya Dataflow, di mana operasi terjadi secara konkuren (bersamaan) melalui penetapan sinyal (signal assignments).

Implementasi pada kode (checksum_unit.vhd): Kami menggunakan adder tree paralel untuk menjumlahkan word 16-bit dari header IPv4 secara serentak.

```
-- Lines 120-132: Adder tree (concurrent dataflow)
-- Level 1: Add pairs of words (5 additions in parallel)
```

```
sum_01 <= ('0' & word0) + ('0' & word1);
sum_23 <= ('0' & word2) + ('0' & word3);
sum_45 <= ('0' & word4) + ('0' & word5);
sum_67 <= ('0' & word6) + ('0' & word7);
sum_89 <= ('0' & word8) + ('0' & word9);
-- Level 4: Final addition (extend to 20 bits)
sum_all <= ('0' & sum_01234567) + ('0' & sum_89_ext2);
```

Kode di atas tidak menggunakan proses atau clock. Sinyal sum_01, sum_23, dst., dihitung secara instan segera setelah nilai input word berubah. Gaya dataflow ini sangat efisien untuk operasi aritmatika kecepatan tinggi karena mengeksploitasi paralelisme perangkat keras FPGA.

2.2.2 Modul 3: Behavioral Description (TTL Unit)

Modul ini berfokus pada perancangan logika sekuensial menggunakan blok process, yang perilakunya dikendalikan oleh sinyal clock dan reset.

Implementasi pada kode (ttl_unit.vhd): Unit ini mengurangi nilai TTL dan mendeteksi apakah paket harus dibuang (drop).

```
ttl_process : process(clk, rst)
    variable ttl_temp : unsigned(7 downto 0);
begin
    if rst = '1' then
        -- Reset registers...
    elsif rising_edge(clk) then
        if enable = '1' then
            -- Extract TTL from header (byte 8 = bits 95:88)
            ttl_temp := unsigned(header_in(95 downto 88));

            -- Check if TTL is 0 or 1 (packet should be dropped)
            if ttl_temp <= 1 then
                drop_reg <= '1';
                ttl_out <= (others => '0');
            else
                drop_reg <= '0';
                ttl_out <= ttl_temp - 1;
            end if;
            -- ...
```

```
        end if;
    end if;
end process ttl_process;
```

Penggunaan process(clk, rst) dan rising_edge(clk) mendefinisikan sirkuit sekuensial (Register). Logika if-else di dalamnya menentukan perilaku unit: jika ttl_temp <= 1, paket ditandai untuk didrop, jika tidak, TTL dikurangi. Perubahan hanya terjadi saat ada tepi naik clock, karakteristik utama desain Behavioral.

2.2.3 Modul 4: Testbench (System Verification)

Modul ini berkaitan dengan pembuatan wadah simulasi untuk memverifikasi fungsionalitas desain tanpa melibatkan perangkat keras fisik, menggunakan teknik Assertion-Based Verification.

Implementasi pada kode (packet_processor_tb.vhd): Testbench menggunakan prosedur untuk menyederhanakan skenario pengujian dan assert untuk pengecekan otomatis.

```
-- Procedure to process a packet and check result
procedure process_packet( ... ) is
begin
    -- ... (Stimulus generation logic) ...

    -- Check drop
    if drop = expect_drop then
        report "  PASS: drop=" & std_logic'image(drop) & "
(expected)" severity note;
    else
        report "  FAIL: drop=" & std_logic'image(drop) &
               " expected=" & std_logic'image(expect_drop)
severity error;
        fail_count <= fail_count + 1;
    end if;
    -- ...
end procedure;

-- Test 2: TTL = 1 (should drop) implementation
```

```
process_packet(test_header, '1', 0, "TTL=1 Expiry Test");
```

Prosedur process_packet mengirimkan sinyal input ke DUT (Device Under Test) dan menunggu output valid. Kemudian, ia membandingkan output drop aktual dengan expect_drop. Jika tidak cocok, simulasi melaporkan error. Ini memungkinkan pengujian skenario kompleks (seperti TTL Expired) dilakukan secara otomatis dan berulang.

2.2.4 Modul 5: Structural Description (Top-Level)

Modul ini berfokus pada penggabungan modul-modul kecil (komponen) menjadi satu sistem yang utuh melalui instansiasi dan pemetaan port (wiring).

Implementasi pada Kode (packet_processor.vhd): Entity top-level menghubungkan Microcontroller, TTL Unit, dan Checksum Unit.

```
-- Microcontroller Instance
u_micro_ctrl : micro_controller
    port map (
        clk          => clk,
        ttl_enable   => ctrl_ttl_en,   -- Sinyal kontrol internal
        cksum_enable => ctrl_cksum_en, -- Sinyal kontrol internal
        -- ...
    );

-- TTL Unit Instance
u_ttl : ttl_unit
    port map (
        clk            => clk,
        enable         => ctrl_ttl_en, -- Terhubung ke output
Microcontroller
        header_in      => header_reg,
        header_ttl_out => ttl_header_out,
        ---
    );
```

Kode ini menunjukkan hierarki desain. Sinyal internal ctrl_ttl_en bertindak sebagai kabel yang menghubungkan port output ttl_enable milik Microcontroller ke port input enable milik TTL Unit. Ini memisahkan logika kontrol dari logika pemrosesan data.

2.2.5 Modul 6: Looping and Generate Statements (Firewall Comparator)

Modul ini memanfaatkan fitur VHDL untuk menghasilkan perangkat keras yang berulang secara otomatis, membuat kode lebih ringkas dan scalable.

Implementasi pada Kode (firewall_rom_checker.vhd): Kami menggunakan FOR-GENERATE untuk membuat 10 komparator alamat IP secara paralel.

```
-- Concurrent Comparator Array (MODULE 6)
gen_comparators: for i in 0 to N_RULES-1 generate
    concurrent_matches(i) <= '1' when ip_reg = BLOCKED_IPS(i) else
'0';
end generate gen_comparators;
```

Sebagai alternatif dari penulisan 10 baris kode pembanding manual blok generate ini secara otomatis mensintesis sirkuit pembanding untuk setiap aturan IP dalam BLOCKED_IPS (dari indeks 0 hingga 9). Jika N_RULES diubah, jumlah perangkat keras akan menyesuaikan secara otomatis saat kompilasi.

2.2.6 Modul 7: Functions and Procedures (Helpers Package)

Modul ini berfokus pada penggunaan subprogram (fungsi) untuk mengenkapsulasi logika kompleks agar dapat digunakan kembali (reusable).

Implementasi pada Kode (helpers.vhd): Fungsi untuk perhitungan aritmatika Ones-Complement dengan penanganan carry.

```
-- FUNCTION: ones_complement_add_fold
function ones_complement_add_fold(
```

```
    a : unsigned(15 downto 0);
    b : unsigned(15 downto 0)
) return unsigned is
    variable sum17  : unsigned(16 downto 0);
    -- ...
begin
    sum17 := ones_complement_add(a, b);
    -- Fold carry back into sum
    sum16 := sum16 + carry;
    return sum16;
end function ones_complement_add_fold;
```

Fungsi ones_complement_add_fold menyembunyikan kompleksitas matematika penjumlahan checksum (di mana carry harus ditambahkan kembali ke LSB). Fungsi ini kemudian dipanggil berkali-kali di dalam checksum_unit.vhd dan testbench, memastikan konsistensi logika di seluruh proyek.

2.2.7 Modul 8: Finite State Machines (Firewall FSM)

Modul ini merancang sistem kontrol sekuensial yang berpindah antar state (status) untuk mengatur alur kerja yang kompleks.

Implementasi pada kode (firewall_rom_checker.vhd): FSM digunakan untuk mengatur urutan pemeriksaan aturan IP.

```
type state_t is (IDLE, CHECKING, DONE_STATE);

-- FSM Process
case state is
    when IDLE =>
        if start = '1' then
            state <= CHECKING;
        end if;

    when CHECKING =>
        if ip_reg = current_rule then
            match_found <= '1';
            state <= DONE_STATE;
```

```
        elsif rule_index >= N_RULES - 1 then
            state <= DONE_STATE; -- Selesai tanpa match
        else
            rule_index <= rule_index + 1; -- Cek aturan berikutnya
        end if;

    when DONE_STATE =>
        state <= IDLE;
end case;
```

FSM ini mengontrol transisi dari menunggu (IDLE), melakukan pengecekan berulang (CHECKING), hingga selesai (DONE_STATE). Logika ini memastikan bahwa Firewall memeriksa aturan satu per satu setiap siklus clock sampai ditemukan kecocokan atau semua aturan diperiksa.

2.2.8 Modul 9: Microprogramming (Micro-Controller)

Modul ini mengimplementasikan Unit Kontrol berbasis instruksi mikro yang tersimpan dalam ROM, menggantikan FSM hardwired yang kaku.

Implementasi pada kode (micro_controller.vhd dan microcode_rom.vhd): Microcontroller membaca instruksi dari ROM dan menerjemahkan bit-bitnya menjadi sinyal kontrol.

```
-- micro_controller.vhd: Decode Logic
ui_ttl_en    <= micro_instr(15); -- Bit 15 mengontrol TTL Unit
ui_cksum_en  <= micro_instr(14); -- Bit 14 mengontrol Checksum
Unit
ui_fw_start  <= micro_instr(13); -- Bit 13 memulai Firewall
ui_wait_fw   <= micro_instr(11); -- Bit 11 mengaktifkan Wait State

-- Logic untuk Wait State (Conditional Branching)
if ui_wait_fw = '1' and fw_done = '0' then
    -- Tahan MPC (Microprogram Counter), jangan lanjut ke
instruksi berikutnya
    mpc_next <= mpc;
else
    -- Lanjut eksekusi
```

```
    mpc_next <= resize(ui_next_addr, ADDR_WIDTH);
end if;
```

Sistem ini membaca instruksi 16-bit dari microcode_rom. Bit spesifik langsung mengaktifkan modul lain (misal Bit 15 mengaktifkan TTL). Fitur ui_wait_fw memungkinkan Microcontroller untuk "berhenti sejenak" (pause) menunggu modul Firewall selesai bekerja sebelum melanjutkan ke instruksi berikutnya. Ini memberikan fleksibilitas tinggi; urutan pemrosesan dapat diubah hanya dengan mengedit isi ROM tanpa merubah rangkaian logika.

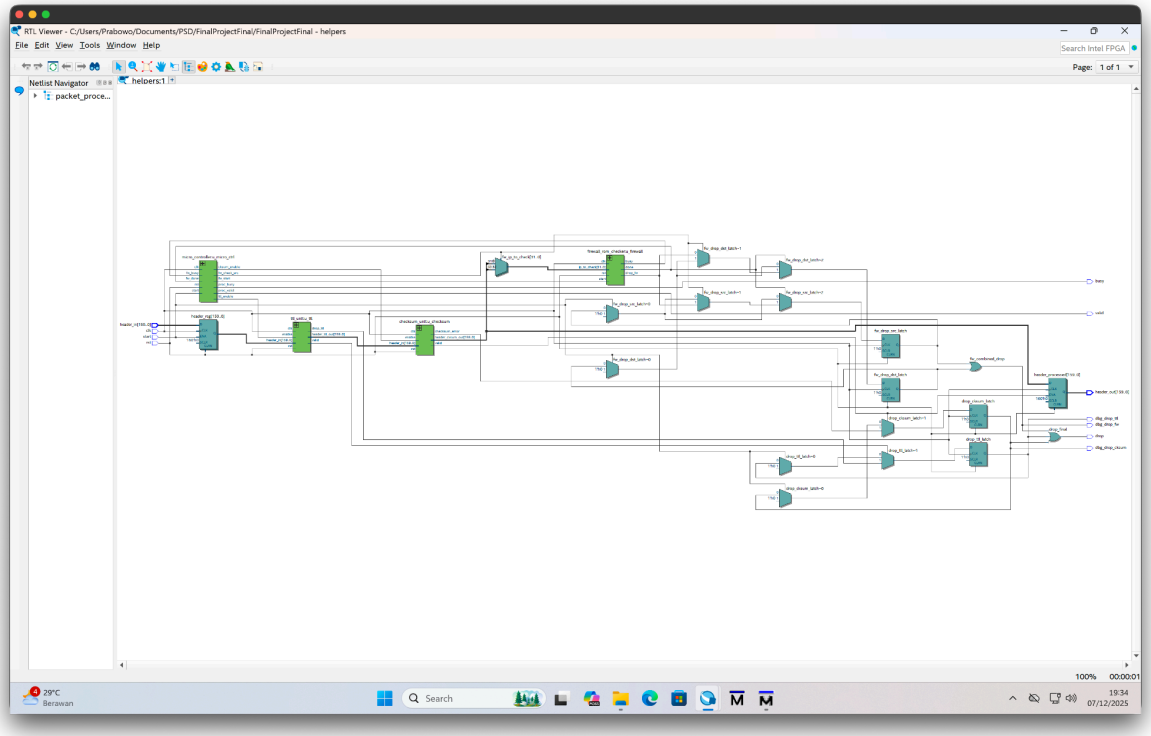


Fig 1. Schematic

Hasil sintesis menunjukkan validasi terhadap desain hierarkis yang didefinisikan dalam Modul 5 (Structural). Alat sintesis berhasil mengenali instansiasi komponen dan mempertahankan batas-batas modul, yang terlihat dari blok-blok hijau terpisah. Aliran data (datapath) terlihat mengalir secara pipelined dari kiri ke kanan: data header masuk ke register input, diproses oleh unit TTL, kemudian diteruskan ke unit Checksum, dan akhirnya diperiksa oleh Firewall. Struktur ini mengonfirmasi bahwa integrasi sistem berjalan sesuai logika desain, di mana Unit Kontrol (Microcontroller) berada di posisi strategis untuk mengirimkan sinyal enable ke seluruh unit pemrosesan tanpa mencampuri aliran data utama.

Secara rinci diagram RTL memvisualisasikan korespondensi langsung antara konstruksi kode VHDL dengan elemen perangkat keras yang terbentuk. Struktur *Multiplexer* (MUX) yang muncul pada jalur input *Firewall* memvalidasi implementasi logika seleksi alamat IP yang bekerja secara kondisional. Di sisi *output*, rangkaian gerbang logika kombinatorial teridentifikasi berfungsi menggabungkan sinyal *drop* dari berbagai unit menjadi satu keputusan final. Selain itu, distribusi register (*flip-flop*) yang teratur di setiap perbatasan modul memvalidasi bahwa sistem beroperasi secara sinkron penuh, memastikan integritas sinyal antar tahap pemrosesan dan membuktikan bahwa desain telah memvisualisasikan desain RTL yang baik tanpa adanya kesalahan timing ataupun latch secara implisit

CHAPTER 3

TESTING AND ANALYSIS

3.1 TESTING

Pengujian sistem IPv4 Packet Processor dilakukan menggunakan metode simulasi functional verification dengan testbench berbasis VHDL. Pengujian ini bertujuan untuk memverifikasi kebenaran logika dari setiap sub-modul (Unit Testing) dan integrasi sistem secara keseluruhan (System Integration Testing).

3.1.1 TESTING METHODOLOGY

Kami menggunakan dua file testbench utama untuk memvalidasi desain: **firewall_checker_tb.vhd**: Berfokus pada pengujian modul Firewall ROM Checker untuk memastikan Finite State Machine (FSM) mendeteksi alamat IP yang diblokir dengan benar. **packet_processor_tb.vhd**: Merupakan top-level testbench yang menguji integrasi antara Mikrokontroler, TTL Unit, Checksum Unit, dan Firewall.

Testbench dirancang menggunakan pendekatan self-checking (Module 4), di mana testbench secara otomatis membandingkan output dari Device Under Test (DUT) dengan nilai yang diharapkan menggunakan prosedur check_ip dan process_packet. Hasil pengujian dilaporkan ke konsol simulator menggunakan fungsi report dan assert.

3.1.2 TESTING SCENARIOS

Skenario pengujian mencakup berbagai kondisi lalu lintas jaringan yang mungkin terjadi:

- **Reset Behavior**: Memastikan seluruh register dan state mikrokontroler kembali ke kondisi awal saat sinyal rst diaktifkan
- **TTL Processing**:

  TTL Expiry: Mengirimkan paket dengan TTL=1. Sistem diharapkan menjatuhkan (drop) paket tersebut karena TTL akan menjadi 0.

  Normal TTL: Mengirimkan paket dengan TTL=64. Sistem diharapkan meneruskan paket dengan TTL yang telah dikurangi menjadi 63.
- **Firewall Filtering**:

  Blocked Source IP: Menguji 10 alamat IP yang terdaftar dalam ROM (misalnya 10.1.0.5). Sistem harus menghasilkan sinyal drop.

  Blocked Destination IP: Memastikan pemblokiran bekerja baik untuk Source maupun Destination IP sesuai instruksi mikrokontroler.

  Allowed IP: Mengirimkan paket dengan IP yang tidak terdaftar (misalnya 10.1.0.7), sistem harus meneruskan paket.
- **Checksum Recomputation**: Memastikan Checksum Unit menghitung ulang header checksum yang valid setelah nilai TTL berubah, bahkan jika checksum input awal salah (korup).

3.2 RESULT

Bagian ini menyajikan bukti visual dari hasil simulasi yang telah dilakukan.

3.2.1 SIMULATION WAVEFORMS

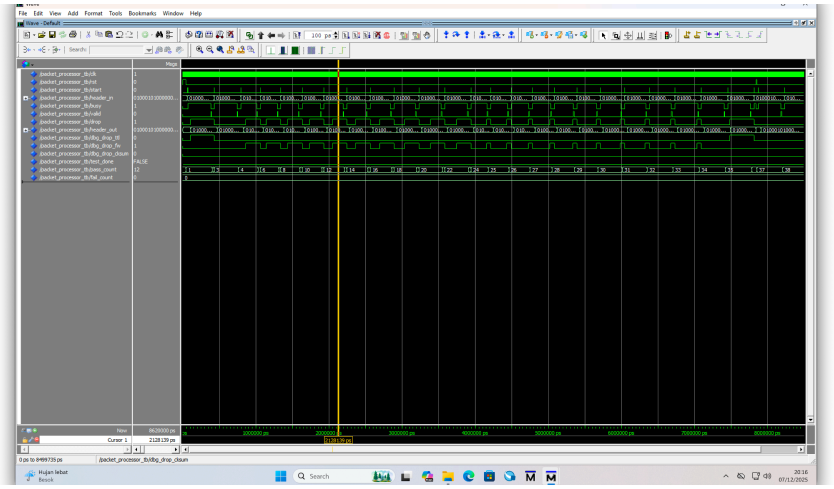
Analisis Waveform: Pada Gambar 3.1, kursor kuning berada pada posisi waktu simulasi di mana sistem sedang memproses paket yang dilarang. Dapat diamati bahwa:

Sinyal header_in diberikan oleh testbench.

Sinyal busy aktif ('1') menandakan mikrokontroler sedang menjalankan instruksi.

Sinyal dbg_drop_fw (baris ke-8 dari atas) bernilai tinggi ('1'). Ini mengindikasikan bahwa Firewall Checker berhasil mendeteksi IP terblokir dan mengirimkan sinyal ke logika kombinatorial utama.

Akibatnya, sinyal drop utama menjadi '1', yang berarti paket tersebut sukses dibuang oleh sistem.



Gambar 3.1 memperlihatkan perilaku sinyal pada level Top Entity.

3.2.2 AUTOMATED TEST REPORT

Analisis Laporan: Transcript pada Gambar 3.2 menunjukkan ringkasan eksekusi testbench hingga waktu simulasi 8615 ns.

Test Coverage: Terlihat pengujian mencakup berbagai kasus sudut (edge cases) seperti "Test 7: TTL=0 (Already Expired)" dan "Test 10: Checksum Recomputation (corrupt input)".

Pass Count: Sistem berhasil melewati 39 skenario pengujian (PASSED: 39) tanpa ada satu pun kegagalan (FAILED: 0).

Pesan "ALL TESTS PASSED!" mengonfirmasi bahwa seluruh logika desain (TTL, Checksum, dan Firewall) berfungsi sesuai spesifikasi yang ditetapkan.

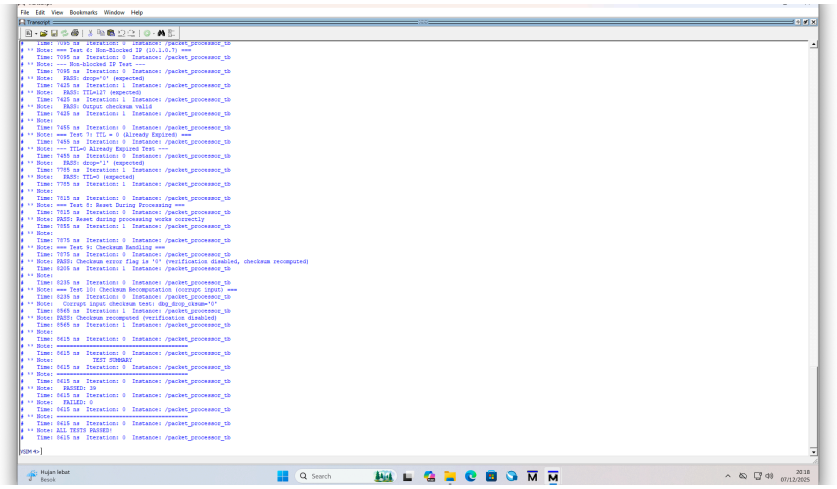


Figure 3.2 ModelSim Transcript showing "ALL TESTS PASSED" summary.

3.3 ANALYSIS

Implementasi ini menunjukkan karakteristik yang sangat sesuai untuk implementasi FPGA dibandingkan mikrokontroler konvensional:

Instruction Efficiency (Module 9): Sistem menggunakan custom microcontroller yang didesain khusus. Tidak seperti prosesor umum yang harus melakukan fetch-decode-execute yang lambat untuk operasi bit, arsitektur ini memproses header IP secara langsung menggunakan instruksi mikro yang dioptimalkan. Berdasarkan waveform (Gambar 3.1), pemrosesan satu paket hanya membutuhkan waktu yang sangat singkat (interval antar sinyal valid), memungkinkan throughput jaringan yang tinggi.

Parallel Dataflow (Module 2): Unit Checksum melakukan penjumlahan 16-bit secara paralel (Module 2 Dataflow) dalam satu siklus. Jika hal ini dilakukan pada mikrokontroler biasa (seperti Arduino), akan dibutuhkan looping software yang memakan puluhan siklus clock.

Deterministic Latency: Penggunaan FSM (Module 8) dan Microcode ROM menjamin bahwa waktu pemrosesan paket selalu deterministik (dapat diprediksi). Hal ini krusial untuk perangkat jaringan agar tidak terjadi jitter atau penundaan yang tidak terduga, sebuah keunggulan mutlak FPGA dibanding sistem berbasis OS/software.

CHAPTER 4

CONCLUSION

- Proyek ini berhasil mengintegrasikan seluruh materi praktikum dari Modul 2 hingga Modul 9 menjadi satu sistem utuh yang fungsional, mencakup teknik dataflow, behavioral, structural, looping, functions, FSM, dan microprogramming.
- Sistem terbukti mampu memproses header paket IPv4 (160-bit) secara efisien dan deterministik menggunakan FPGA, mengatasi keterbatasan latensi dan throughput yang umum ditemukan pada solusi berbasis perangkat lunak konvensional.
- Desain sistem memisahkan jalur data (Datapath) yang terdiri dari unit pemrosesan (TTL, Checksum, Firewall) dari unit kontrol (Control Unit) berbasis mikrokontroler. Pemisahan ini divalidasi oleh hasil sintesis RTL Viewer yang menunjukkan struktur hierarkis yang jelas dengan interkoneksi sinyal kontrol yang terpisah dari bus data utama.
- Penggunaan custom microcontroller dengan instruksi mikro dalam ROM (Modul 9) memberikan fleksibilitas tinggi dalam mengatur urutan eksekusi perangkat keras yang kompleks tanpa perlu merubah rangkaian logika, serta memastikan latensi pemrosesan yang dapat diprediksi.
- Implementasi Unit Checksum menggunakan pendekatan parallel dataflow (Modul 2) memungkinkan perhitungan penjumlahan ones-complement untuk 10 word 16-bit diselesaikan dalam satu siklus waktu yang sangat singkat, jauh lebih cepat dibandingkan pendekatan looping sekuensial pada mikrokontroler standar.
- Modul Firewall Statis (Modul 6 & 8) berhasil menerapkan pemblokiran paket berdasarkan daftar 10 alamat IP yang disimpan dalam ROM. Penggunaan FOR-GENERATE untuk komparator paralel dan FSM untuk kontrol sekuensial terbukti efektif mendeteksi dan membuang paket terlarang secara akurat.
- Keputusan akhir untuk membuang paket (drop) diimplementasikan secara robust melalui logika kombinasional yang menggabungkan hasil deteksi dari TTL expiry, kesalahan checksum, dan pemblokiran firewall. Hal ini memastikan tidak ada paket cacat atau berbahaya yang lolos dari sistem.

- Pengujian sistem menggunakan metode Assertion-Based Verification (Modul 4) dengan testbench otomatis terbukti sangat efektif. Sistem berhasil melewati 39 skenario pengujian (ALL TESTS PASSED) yang mencakup berbagai kasus sudut (corner cases) seperti TTL=1, checksum korup, dan berbagai alamat IP blokir.
- Hasil sintesis RTL Viewer mengonfirmasi bahwa kode VHDL yang ditulis memenuhi kaidah Register Transfer Level (RTL) yang baik dan synthesizable. Implementasi fisik menunjukkan penggunaan register (flip-flop) yang tepat untuk sinkronisasi sinyal dan tidak adanya latch implisit yang dapat menyebabkan kesalahan timing.
- Penggunaan parameter generik (seperti N_RULES) dan konstruksi generate dalam kode VHDL memungkinkan sistem ini untuk dengan mudah diskalakan. Menambah jumlah aturan firewall atau mengubah parameter sistem dapat dilakukan hanya dengan mengubah nilai konstanta tanpa perlu menulis ulang logika inti

REFERENCES

[1] GeeksforGeeks, “VHDL - Very High Speed Integrated Circuit Hardware Description Language,” GeeksforGeeks, Nov. 2025. https://www.geeksforgeeks.org/electronics-engineering/vhdl-very-high-speed-integrated-circuit-hardware-description-language/ (accessed Dec. 7, 2025).

[2] GeeksforGeeks, “What is Internet Protocol Version 4 (IPv4),” GeeksforGeeks, Oct. 2025. https://www.geeksforgeeks.org/computer-networks/what-is-ipv4/ (accessed Dec. 7, 2025).

[3] Russell, “Tutorial - What is a Testbench (simulation),” Nandland, Jun. 09, 2022. https://nandland.com/what-is-a-testbench/ [Accessed: Sep. 15, 2025]

[4] “VHDL Structural Modeling Style,” Surf-VHDL, Oct. 19, 2017. https://surf-vhdl.com/vhdl-syntax-web-course-surf-vhdl/vhdl-structural-modeling-style

[5] J. J. Jensen, “How to use Port Map instantiation in VHDL,” VHDLwhiz, Sep. 18, 2017. https://vhdlwhiz.com/port-map/ (accessed Sep. 29, 2025)

[6] Russell, “Generics in VHDL,” Nandland, Jun. 18, 2022. https://nandland.com/generics-in-vhdl/ (accessed Sep. 29, 2025).

[7] John, “Writing Reusable VHDL Code using Generics and Generate Statements,” FPGA Tutorial, May 30, 2020. https://fpgatutorial.com/vhdl-generic-generate/ (accessed Sep. 29, 2025)

[8] “VHDL GUIDELINES FOR SYNTHESIS.” Available: https://web02.gonzaga.edu/faculty/talarico/ee406/20162017/VHDL/vhdl-for-synthesis.pdf [Accessed: Oct. 6, 2025]

[9] GeeksforGeeks, “IPv4 Datagram Header,” GeeksforGeeks, Jul. 03, 2015. https://www.geeksforgeeks.org/computer-networks/introduction-and-ipv4-datagram-header/

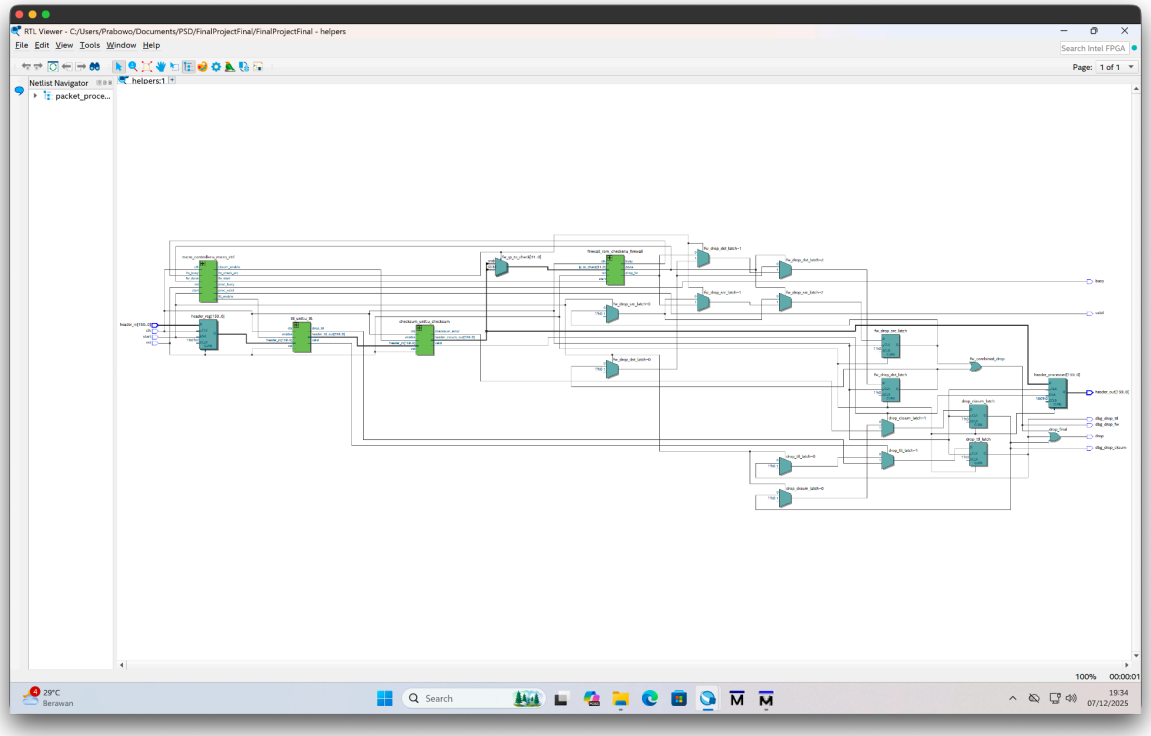
[10] R. Braden, D. Borman, and C. Partridge, "Computing the Internet Checksum," IETF Request for Comments 1071, Sep. 1988. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc1071 [Accessed: Dec. 7, 2025].

[11] Information Sciences Institute, "Internet Protocol," IETF Request for Comments 791, Sep. 1981. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc791 [Accessed: Dec. 7, 2025].

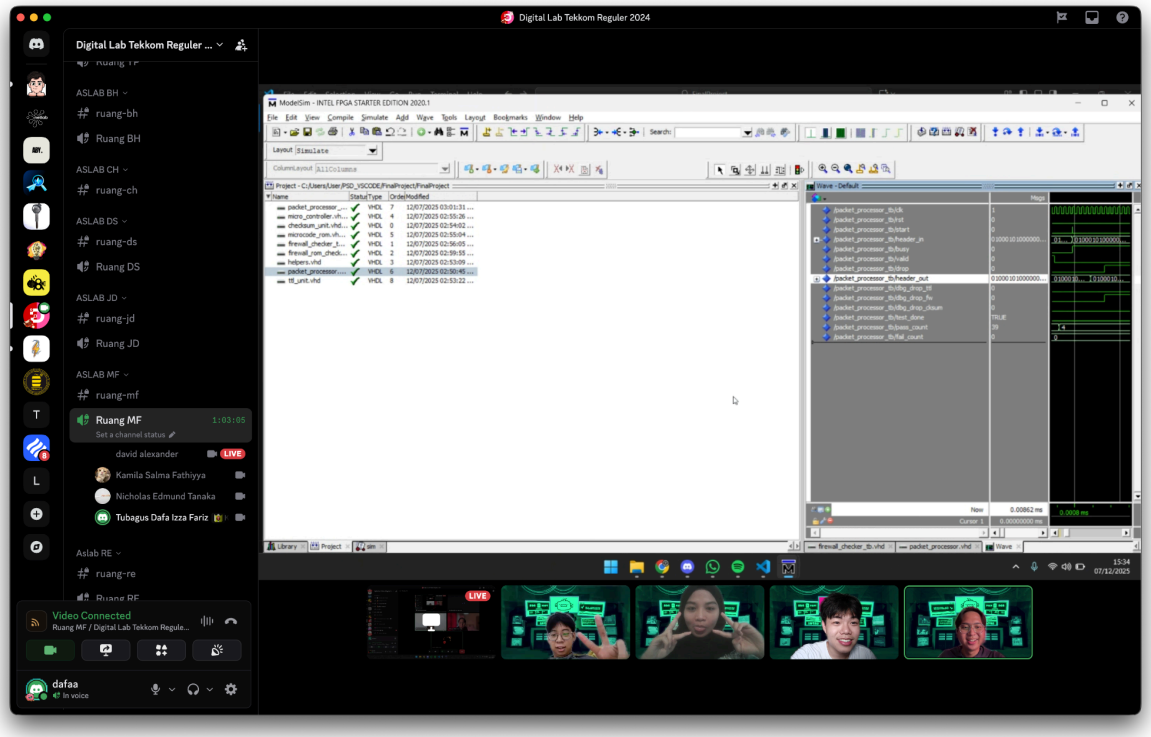
APPENDICES

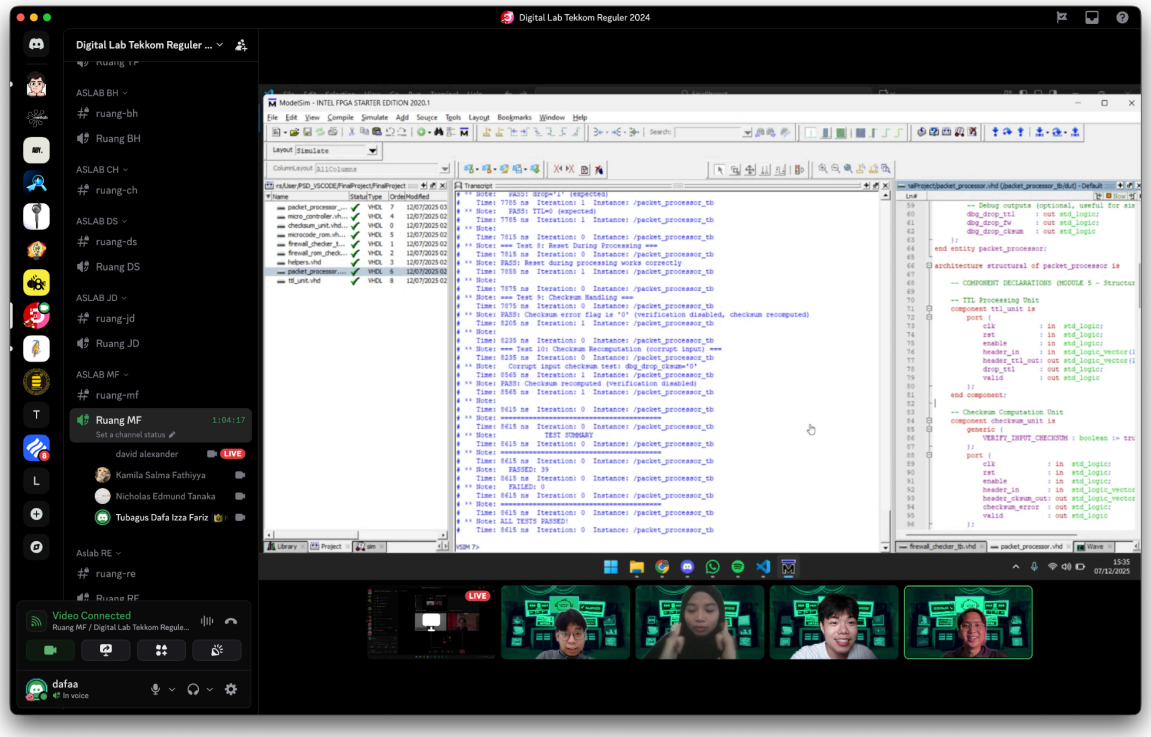
Appendix A: Project Schematic

packet_processor :



Appendix B: Documentation


Digital Lab Tekkom Reguler 2024
Digital Lab Tekkom Reguler ...
ASLAB BH
ruang-bh
Ruang BH
ASLAB CH
ruang-ch
ASLAB DS
ruang-ds
Ruang DS
ASLAB JD
ruang-jd
Ruang JD
ASLAB MF
ruang-mf
Ruang MF
1:03:05
Set a channel status
david alexander
LIVE
Kamila Salma Fathiyya
Nicholas Edmund Tanaka
Tubagus Dafa Izza Fariz
Aslab RE
ruang-re
Video Connected
Ruang MF / Digital Lab Tekkom Regule...
dafaa
In voice
ModelSim - INTEL FPGA STARTER EDITION 2020.1
15:34
07/12/2025


Digital Lab Tekkom Reguler 2024
Digital Lab Tekkom Reguler ...
ASLAB BH
ruang-bh
Ruang BH
ASLAB CH
ruang-ch
ASLAB DS
ruang-ds
Ruang DS
ASLAB JD
ruang-jd
Ruang JD
ASLAB MF
ruang-mf
Ruang MF
1:04:17
Set a channel status
david alexander
LIVE
Kamila Salma Fathiyya
Nicholas Edmund Tanaka
Tubagus Dafa Izza Fariz
Aslab RE
ruang-re
Video Connected
Ruang MF / Digital Lab Tekkom Regule...
dafaa
In voice
ModelSim - INTEL FPGA STARTER EDITION 2020.1
15:35
07/12/2025