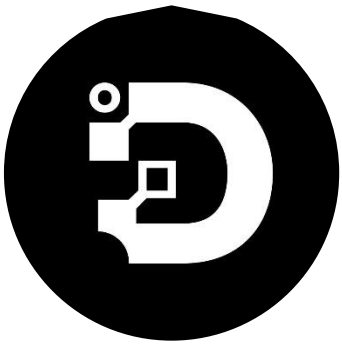


LAPORAN PROYEK PERANCANGAN SISTEM DIGITAL
CALON ASISTEN LABORATORIUM DIGITAL
UNIVERSITAS INDONESIA

PacketFilter – Extended ACL Module

Disusun Oleh:
Muhammad Riyan Satrio Wibowo (RE)
23063392323

1. ABSTRAK

Pada era digital saat ini, internet merupakan hal yang krusial dalam kehidupan sehari-hari. Internet menjadi media untuk mendapatkan informasi dari seluruh dunia. Akan tetapi, seiring berkembangnya internet, semakin banyak pula ancaman yang menunggu di luar. Oleh karena itu, semakin penting pula bagi kita untuk membatasi *traffic* internet mana saja yang boleh masuk dan keluar jaringan.

Access List Control (ACL) merupakan sekumpulan aturan yang didefinisikan untuk mengontrol traffic jaringan dan mengurangi serangan yang terjadi pada jaringan. **Extended ACL** merupakan jenis ACL yang dapat memfilter traffic jaringan berdasarkan IP address sumber dan tujuan, port number sumber dan tujuan, serta protokol yang digunakan. Hal ini memungkinkan administrator untuk mengatur secara spesifik jenis packet data mana yang boleh keluar dan masuk jaringan.

Proyek "**PacketFilter – Extended ACL Module**" bertujuan untuk mengimplementasikan sistem Extended ACL menggunakan bahasa VHDL. Modul yang dibuat akan menerima packet data, kemudian menerapkan aturan yang tertulis pada ACL, dan menentukan apakah packet data tersebut diizinkan oleh jaringan.

2. METODE

a. Arsitektur Proyek

Secara umum, proyek yang dibuat terbagi menjadi 3 lapisan. Lapisan teratas merupakan komponen **acl_system.** Komponen tersebut bertanggungjawab dalam menyimpan aturan-aturan ACL dan menjalankannya satu-persatu menggunakan lapisan kedua, yaitu **top_level**.

Komponen tersebut berfungsi untuk menerapkan satu aturan ACL terhadap input packet data. Komponen ini menghubungkan 2 komponen yang terdapat pada lapisan ketiga, yaitu **microinstr_gen** dan **fsm_interpreter**.

Komponen **microinstr_gen** berfungsi untuk membuat microinstruction berdasarkan parameter yang dikirimkan melalui top_level, kemudian mengembalikan microinstruction tersebut ke top_level. Microinstruction tersebut kemudian akan dijalankan oleh **fsm_interpreter**. Setelah seluruh instruksi dijalankan, maka akan didapatkan keputusan untuk mengizinkan/menolak packet data yang diinput.

b. Penerapan Microprogramming

1) Access List Entry (acl_rule)

Permit (168)	Allowed Src IP (167:136)	Src Mask (135:104)	Allowed Dst IP (103:72)	Dst Mask (71:40)	Allowed Protocol (39:32)	Allowed Src Port (31:16)	Allowed Dst Port (15:0)

Access List Entry (169-bit) berisi parameter untuk setiap aturan pada ACL. Bit permit digunakan untuk mengatur command ACL, yaitu 0 untuk "deny" dan 1 untuk "permit." Bit tersebut diikuti oleh IP address sumber yang diizinkan (32-bit), subnet mask sumber (32-bit), IP address tujuan yang diizinkan (32-bit), subnet mask tujuan (32-bit), protokol yang diizinkan (8-bit), port number sumber yang diizinkan (16-bit), dan port number tujuan yang diizinkan (16-bit).

2) Microinstruction (microinstr)

Opcode (15:12)	Field (11:9)	Permit (8)	Value/JMP Addr (7:0)

a) Opcode (4-bit)

Terdapat 5 instruksi yang dapat digunakan, antara lain:

- 0001 (CMP_IP)

  Digunakan untuk membandingkan data IP address dengan data yang ada pada Value/JMP Addr. Jika nilainya sama, maka cmp_flag akan diset

- 0010 (CMP_PR)

  Digunakan untuk membandingkan data protokol/port number dengan data yang ada pada Value/JMP Addr. Jika nilainya sama, maka cmp_flag akan diset.

- 1000 (JNZ)

  Jika cmp_flag bernilai '0' (data yang dicompare tidak sama), program akan melakukan jump ke instruksi yang berada pada address

Value/JMP Addr. Hal ini dilakukan dengan set flag jump_en dan mengouput nilai pada Value/JMP Addr ke jump_addr.

- 1110 (MATCH)
  Jika Permit bernilai '0' (deny), maka flag drop akan diset. Jika Permit bernilai '1' (permit), maka flag accept akan diset.

- 1111 (NOT_MATCH)
  Flag drop dan accept diclear, menandakan bahwa aturan ACL tidak berlaku pada packet data input.

b) Field (3-bit)

Digunakan untuk melakukan parsing pada packet data input. Berikut adalah logika parsing berdasarkan Field (index) pada **function get_octet**:

```
case index is
    when 0 => return sip(31 downto 24);
    when 1 =>
        case opcode is
            when "0001" => return sip(23 downto 16);
            when "0010" => return pr(7 downto 0);
        end case;
    when 2 =>
        case opcode is
            when "0001" => return sip(15 downto 8);
            when "0010" => return sport(15 downto 8);
        end case;
    when 3 =>
        case opcode is
            when "0001" => return sip(7 downto 0);
            when "0010" => return sport(7 downto 0);
        end case;
    when 4 =>
        case opcode is
            when "0001" => return dip(31 downto 24);
            when "0010" => return dport(15 downto 8);
```

```
        end case;
    when 5 =>
        case opcode is
            when "0001" => return dip(23 downto 16);
            when "0010" => return dport(7 downto 0);
        end case;
    when 6 => return dip(15 downto 8);
    when 7 => return dip(7 downto 0);
end case;
```

Dimana:

- sip : Source IP Address
- dip : Destination IP Address
- sport : Source port number
- dport : Destination port number
- pr : Protocol

c) Permit (1-bit)

Menunjukkan command yang diterapkan pada aturan ACL. Nilai '1' berarti "permit" dan nilai '0' berarti "deny."

d) Value/JMP Addr (8-bit)

Menyimpan nilai yang dibandingkan dengan data pada instruksi CMP_IP (IP Address) dan CMP_PR (Protocol/Port Number). Menyimpan address instruksi tujuan pada instruksi JNZ.

c. Implementasi Modul

1) Microinstruction Generator (microinstr_gen.vhd)

Komponen dengan **style behavioral** ini bekerja sebagai instruction register sekaligus instruction generator. Komponen ini menerima input berupa:

- permit : Command aturan ACL (permit/deny)
- address : Address dari instruksi yang diminta
- allowed_protocol : Protokol yang diizinkan
- allowed_src_ip : IP address sumber yang diizinkan

- allowed_dst_ip : IP address tujuan yang diizinkan
- allowed_src_port : Port number sumber yang diizinkan
- allowed_dst_port : Port number tujuan yang diizinkan

Komponen ini kemudian akan mengoutput microinstruction berdasarkan ketujuh input tersebut kepada komponen top_level. Berikut adalah urutan dari instruksi yang dijalankan jika dituliskan dalam bahasa assembly yang disederhanakan:

```
CMP_IP src_addr(31:24), allowed_src_addr(31:24)
JNZ no_match
CMP_IP src_addr(23:16), allowed_src_addr(23:16)
JNZ no_match
CMP_IP src_addr(15:8), allowed_src_addr(15:8)
JNZ no_match
CMP_IP src_addr(7:0), allowed_src_addr(7:0)
JNZ no_match

CMP_IP dst_addr(31:24), allowed_dst_addr(31:24)
JNZ no_match
CMP_IP dst_addr(23:16), allowed_dst_addr(23:16)
JNZ no_match
CMP_IP dst_addr(15:8), allowed_dst_addr(15:8)
JNZ no_match
CMP_IP dst_addr(7:0), allowed_dst_addr(7:0)
JNZ no_match

CMP_IP protocol, allowed_protocol
JNZ no_match

CMP_IP src_port(15:8), allowed_src_port(15:8)
JNZ no_match
CMP_IP src_port(7:0), allowed_src_port(7:0)
JNZ no_match

CMP_IP dst_port(15:8), allowed_dst_port(15:8)
JNZ no_match
CMP_IP dst_port(7:0), allowed_dst_port(7:0)
JNZ no_match

MATCH
no_match:
NO_MATCH
```

2) **Microinstruction Interpreter (fsm_interpreter)**

Komponen ini berfungsi untuk menjalankan microinstruction yang dihasilkan oleh microinstr_gen menggunakan konsep **microprogramming** pada input packet data yang sudah diparsing. Komponen ini menerapkan **FSM** untuk menjalankan microinstruction secara bertahap.

Terdapat beberapa state pada FSM, antara lain:

a) IDLE

Pada state ini, seluruh flag (jump_en, accept, drop, cmp_flag) dan control signal (done) diclear. Kemudian, program akan menunggu sinyal start dari top_level untuk berpindah ke state selanjutnya. Digunakan pula control signal (control) untuk memberikan delay 1 clock cycle untuk keperluan sinkronisasi.

b) DECODE

Pada state ini, microinstruction diparsing sesuai dengan field yang ada pada bagian *Penerapan Microprogramming*, yaitu dibagi menjadi Opcode, Field, Permit, dan Value/JMP Addr.

c) EXECUTE

Pada state ini, microinstruction dieksekusi sesuai dengan penjelasan pada bagian *Penerapan Microprogramming*. Pada instruksi CMP_IP dan CMP_PR, digunakan **function** get_octet untuk mengambil byte data dari packet data input sesuai dengan nilai pada Field.

d) FINISH

Pada state ini, control signal done akan diset dan program akan kembali ke state IDLE.

3) **Access List Entry Processor (top_level.vhd)**

Komponen ini merupakan komponen inti dari proyek ini, dimana komponen ini berfungsi untuk menjalankan satu aturan ACL (biasa disebut ACE).

Komponen ini menghubungkan dua komponen layer terbawah, yaitu fsm_interpreter dan microinstr_gen dengan **structural programming** menggunakan port mapping. Komponen ini menerima input packet data (input_packet) dan aturan ACL (acl_rule) dari acl_system, kemudian mengirimkan sinyal accept dan drop serta control signal finish.

Seperti fsm_interpreter, komponen ini juga menerapkan **FSM**. Terdapat beberapa state pada FSM, antara lain:

a) IDLE

Pada state ini, seluruh flag (accept, drop) dan control signal (finish) diclear. Instruction address juga direset menjadi 0 agar program kembali ke instruksi pertama. Digunakan pula control signal (control) untuk memberikan delay 2 clock cycle untuk keperluan sinkronisasi.

b) DECODE

Pada state ini, acl_rule diparsing sesuai dengan field yang ada pada bagian *Penerapan Microprogramming*, yaitu dibagi menjadi Permit, Allowed Src IP, Src Mask, Allowed Dest IP, Dest Mask, Allowed Protocol, Allowed Src Port, dan Allowed Dest Port.

c) GEN

Pada state ini, program melakukan masking pada IP address dengan subnet masknya. Selain itu, jika allowed protocol, allowed source port, dan allowed destination port bernilai 0 (“any” dalam ACL), maka protocol, source port, dan destination port input dimasking dengan 0.

d) EXECUTE

Pada state ini, program mengirimkan microinstruction dari microinstr_gen kepada fsm_interpreter. Program kemudian menunggu instruksi selesai dijalankan oleh fsm_interpreter (done = ‘1’). Setelah itu, program akan akan menentukan address instruksi sebelumnya. Jika flag jump_en bernilai ‘0’, program akan mengincrement address instruksi.

Jika flag jump_en bernilai ‘1’, program akan mengambil jump_addr sebagai instruksi selanjutnya. Jika address instruksi sudah mencapai address instruksi terakhir, top_level akan mengirimkan control signal finish kepada acl_system, dan state kembali ke IDLE.

4) ACL System (acl_system.vhd)

Komponen ini merupakan lapisan teratas dari sistem, dimana komponen ini terhubung dengan top_level dengan **structural programming** menggunakan port map. Pada komponen ini, terdapat **ACL_ROM** yang menyimpan sekumpulan aturan ACL yang akan dijalankan pada masing-masing packet data input. Komponen ini kemudian mengirimkan aturan ACL satu persatu berdasarkan index kepada top_level untuk kemudian diproses.

Komponen ini juga menerapkan **FSM**. Terdapat beberapa state pada FSM, antara lain:

a) IDLE

Pada state ini, flag permit, control signal acl_done, dan rule_index direset menjadi 0. Kemudian, program beralih ke state CHECK_RULE.

b) CHECK_RULE

Pada state ini, program akan mengirimkan aturan ACL kepada top_level, kemudian menunggu aturan tersebut selesai diproses (finish = ‘1’). Kemudian, program akan mengirimkan aturan ACL selanjutnya secara sekuensial dengan menginkremen rule_index. Jika rule_index sudah menyentuh aturan terakhir, atau jika terdapat kecocokan pertama antara packet data dan ACE (drop = ‘1’ atau accept = ‘1’), maka program akan meneruskan flag accept ke output permit dan beralih ke state FINISHED.

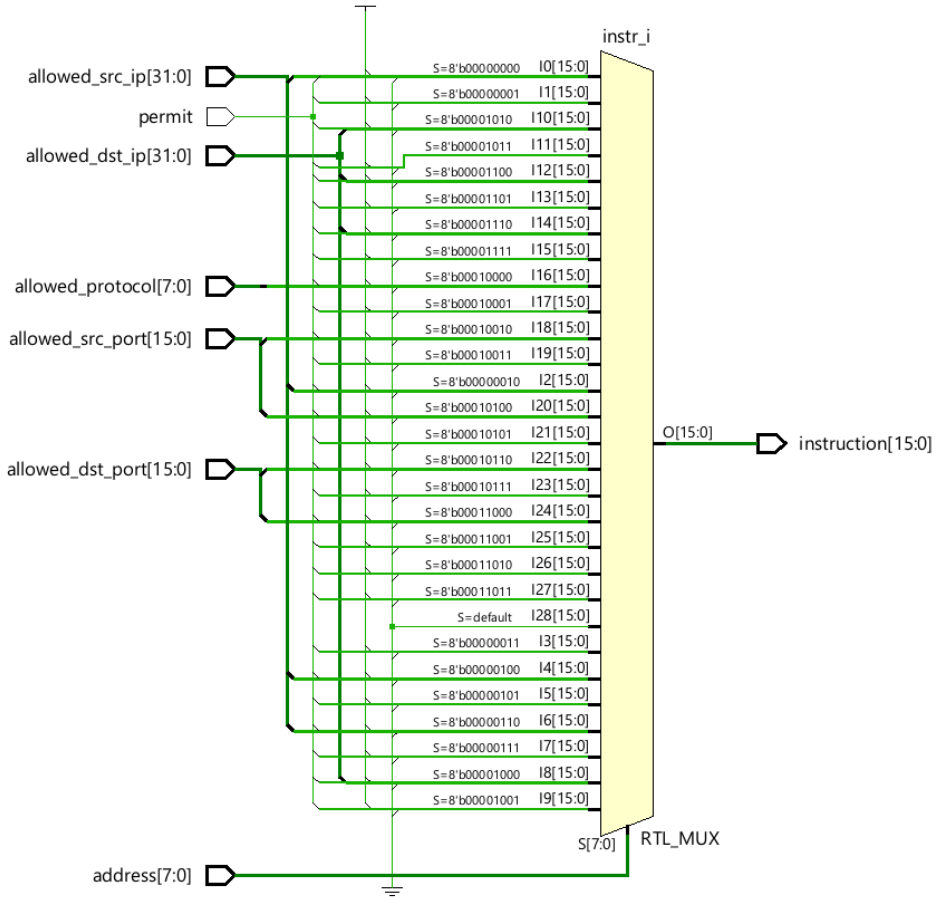
c) FINISHED

Pada state ini, control signal acl_done akan diset dan program kembali ke state IDLE.

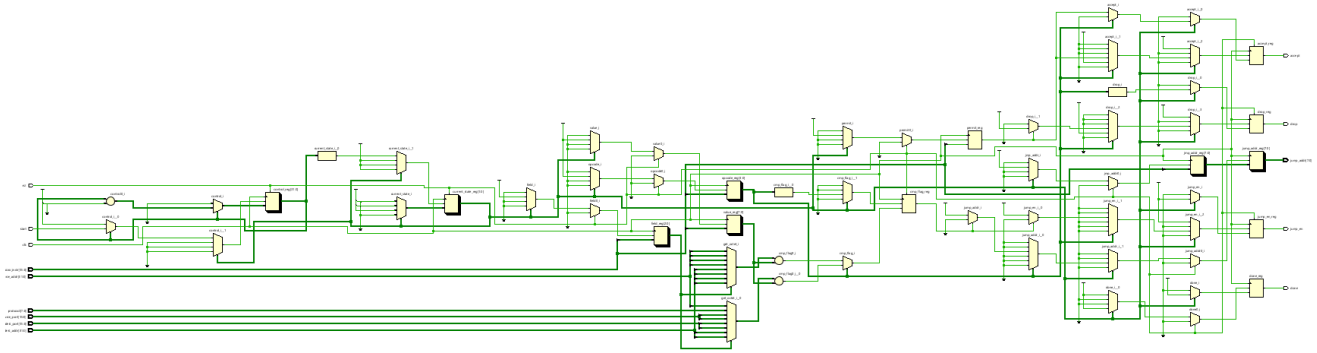
3. DOKUMENTASI

a. Hasil Sintesis Rangkaian

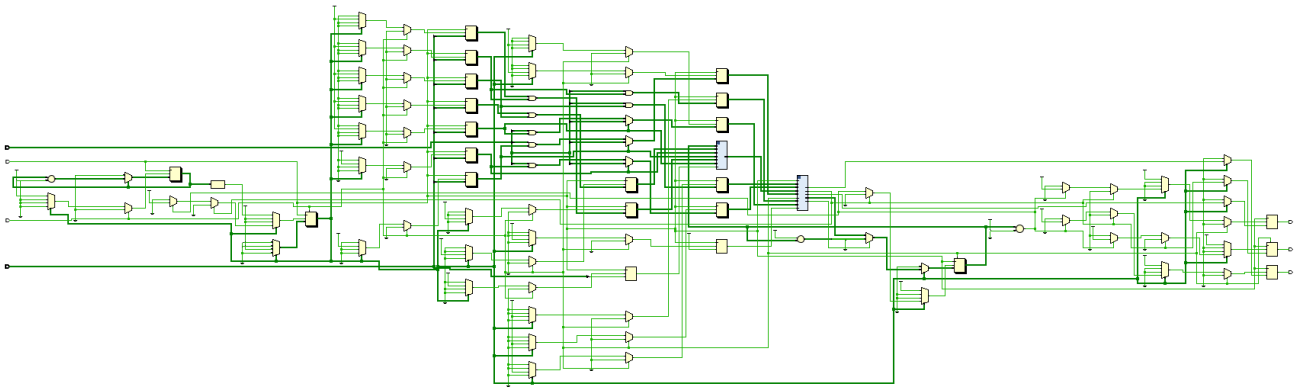
1) Microinstruction Generator (microinstr_gen)



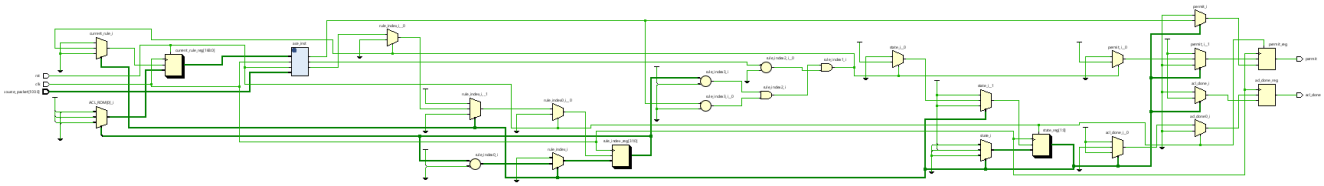
2) Microinstruction Interpreter (fsm_interpreter)



3) Access List Entry Processor (top_level)



4) ACL System (acl_system)



b. Pengujian Sistem

Pengujian sistem dilakukan pada lapisan teratas, yaitu ACL System. Aturan ACL (ACE) dapat diubah sesuai dengan kebutuhan secara langsung pada ACL_ROM. Pada pengujian ini, digunakan 3 aturan ACL, antara lain sebagai berikut:

1) permit tcp 192.168.1.0 0.0.0.255 host 10.20.30.40 eq 80
2) deny ip host 172.16.10.10 any
3) permit icmp any any

Perlu diketahui, input pada sistem merupakan simulasi dari header IPv4 dan juga header TCP, dimana kedua parameter tersebut disatukan menjadi 1 input data. Berikut adalah detail mengenai input pada sistem:

Protocol (103:96)	Source IP (95:64)	Source Port (63:48)	Dest IP (47:16)	Dest Port (15:0)

1) **Testcase 1 (Input Match dengan ACE Pertama)**

Input : 0x06C0A80132C0010A141E280050

Protocol : 06 (TCP)

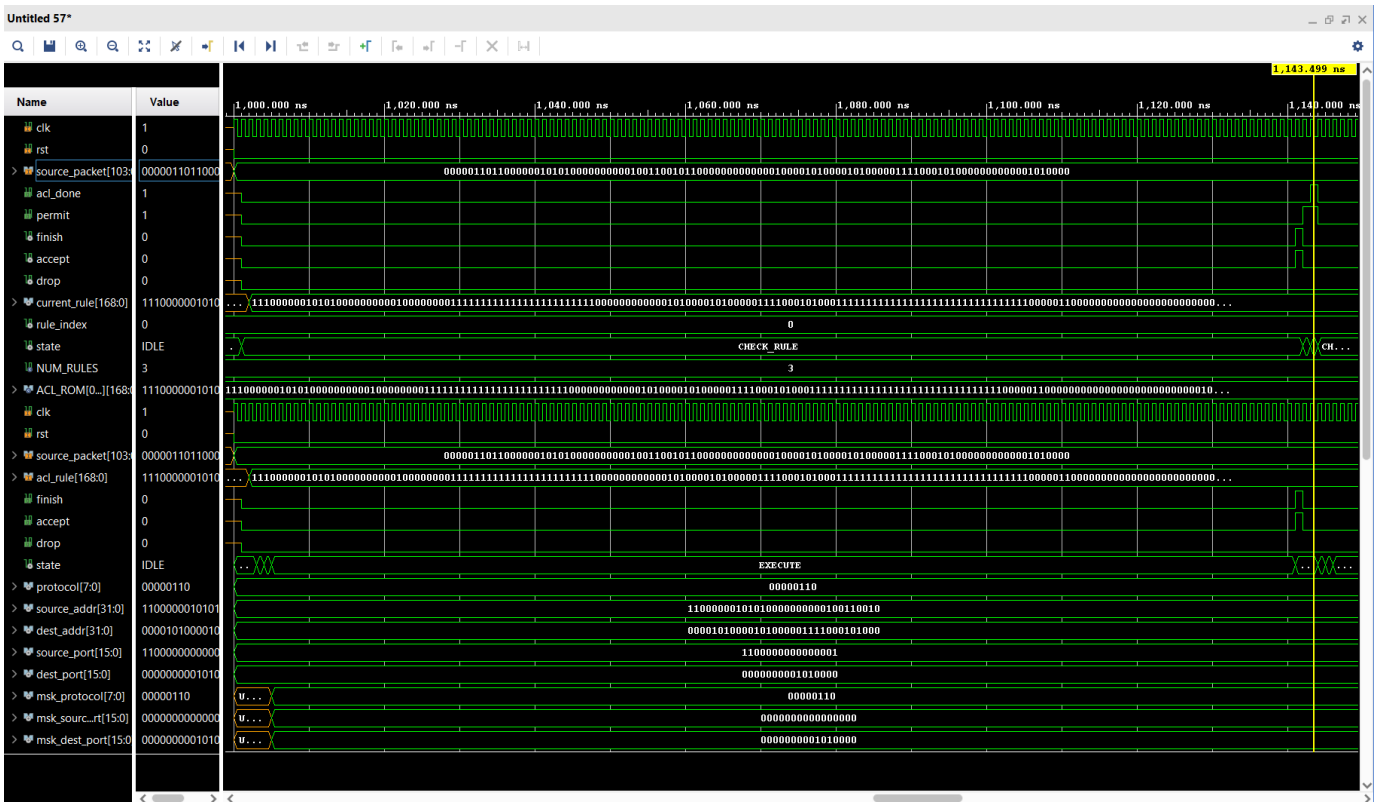
Src IP : 192.168.1.50

Src Port : 49153

Dest IP : 10.20.30.40

Dest Port : 80 (HTTP)

Waveform:



Pada test case tersebut, dapat dilihat bahwa output permit bernilai ‘1’ pada akhir pengecekan rule_index 0 (aturan ACL pertama), sehingga hasil pengujian sesuai.

2) **Testcase 2 (Input Match dengan ACE Kedua)**

Input : 0x11AC100A0AA001080808080035

Protocol : 17 (UDP)

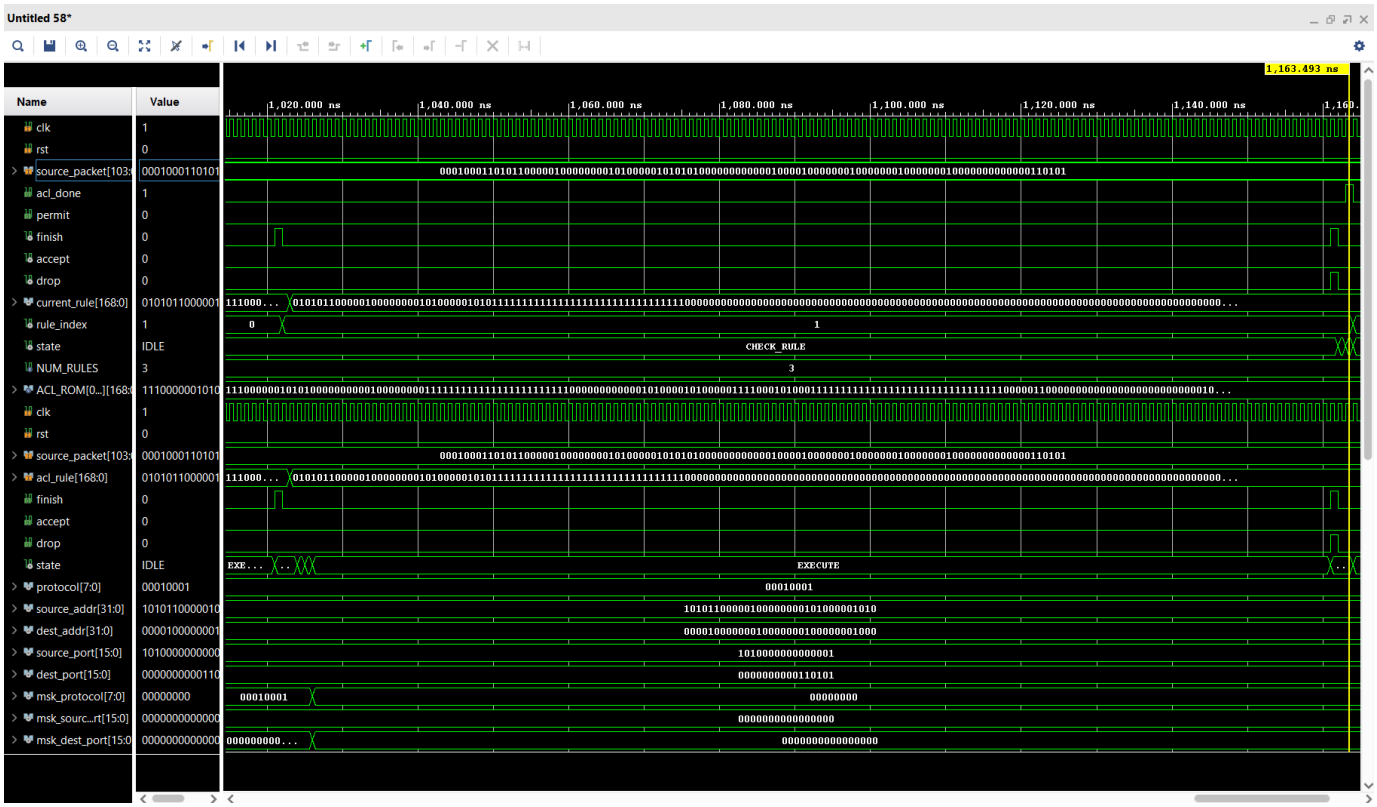
Src IP : 172.16.10.10

Src Port : 40961

Dest IP : 8.8.8.8

Dest Port : 53 (DNS)

Waveform:



Pada test case tersebut, dapat dilihat bahwa output permit bernilai ‘0’ pada akhir pengecekan rule_index 1 (aturan ACL kedua), sehingga hasil pengujian sesuai.

3) Testcase 3 (Input Match dengan ACE Ketiga)

Input : 0x01C0A8016400000A141E280000

Protocol : 01 (ICMP)

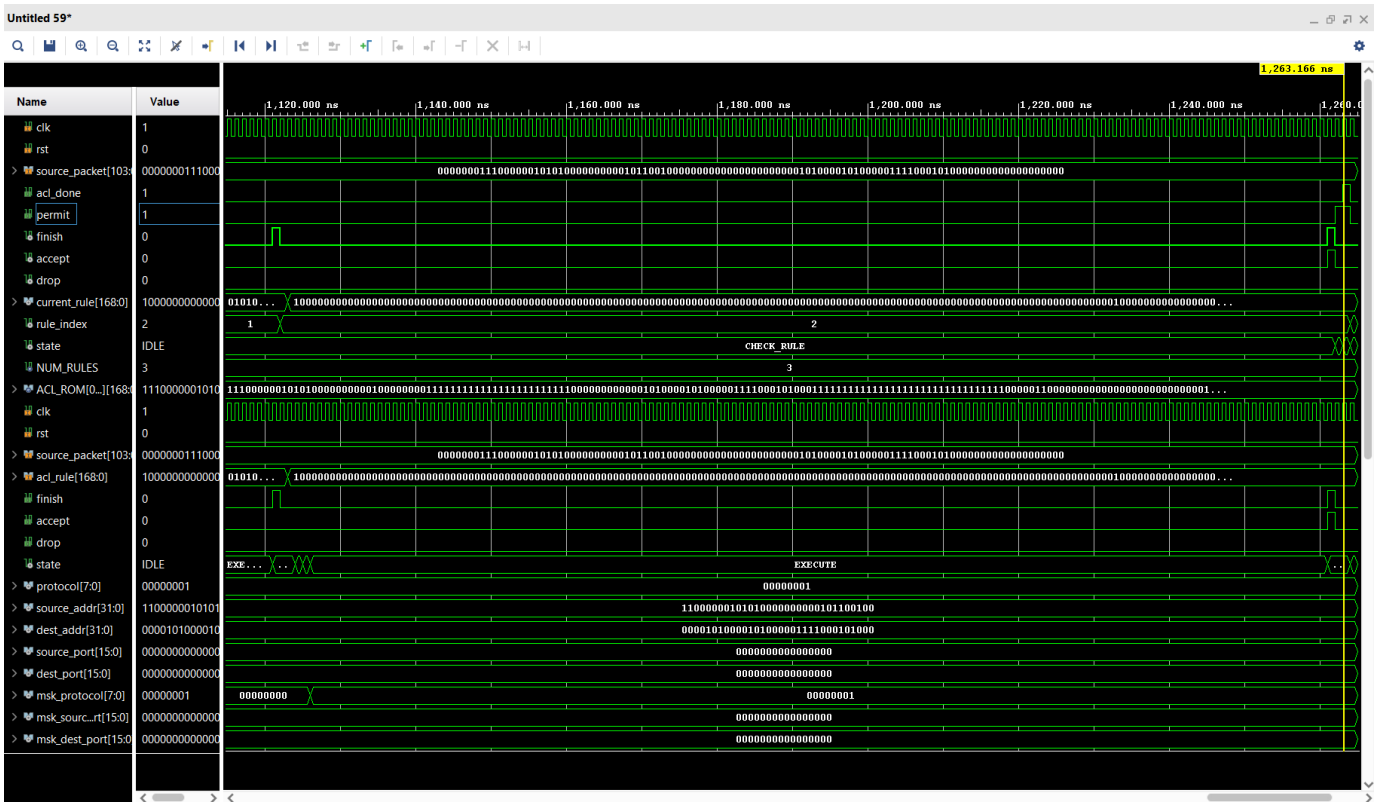
Src IP : 192.168.1.100

Src Port : 0000

Dest IP : 10.20.30.40

Dest Port : 0000

Waveform:



Pada test case tersebut, dapat dilihat bahwa output permit bernilai ‘1’ pada akhir pengecekan rule_index 2 (aturan ACL ketiga), sehingga hasil pengujian sesuai.

4) Testcase 4 (Input Tidak Match dengan ACE Manapun)

Input : 0x1101010101B00102020202B002

Protocol : 17 (UDP)

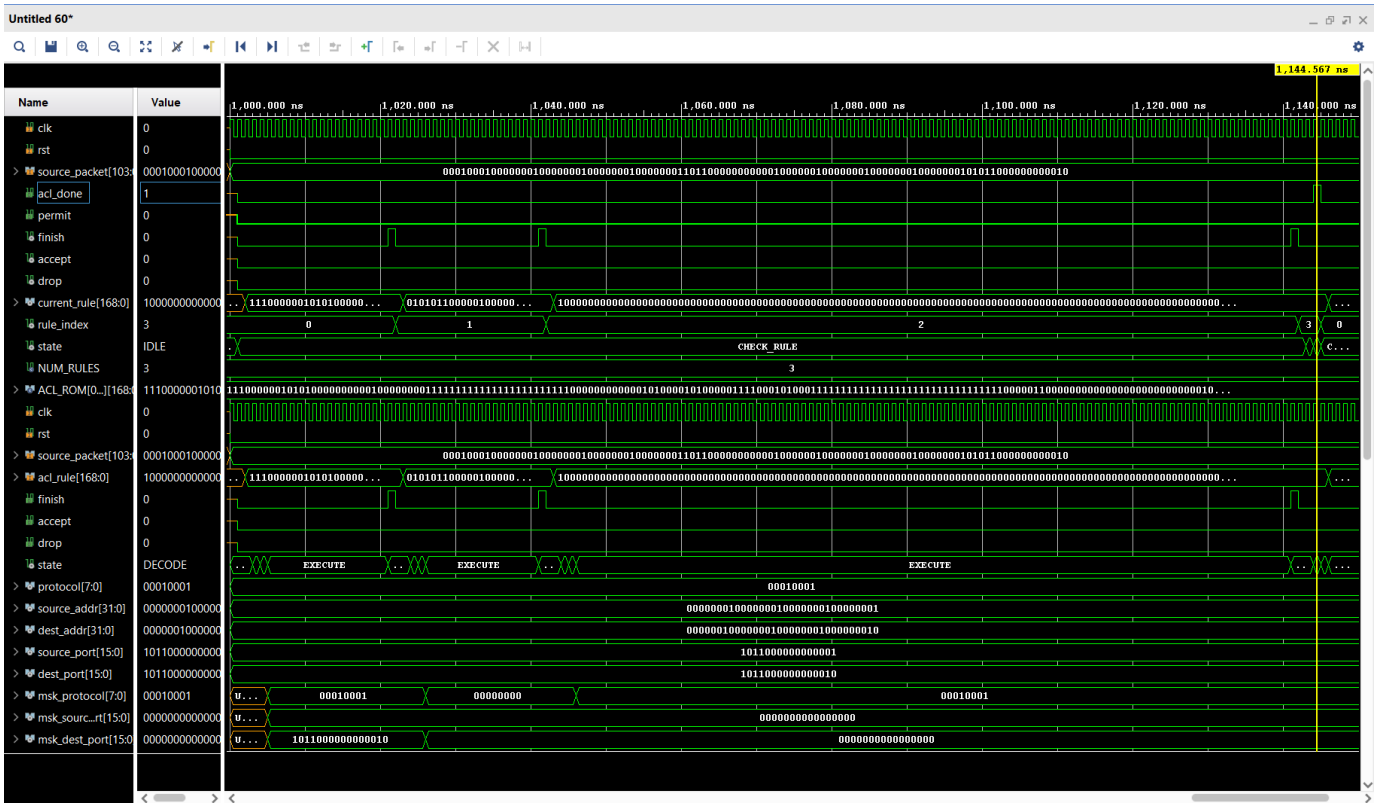
Src IP : 1.1.1.1

Src Port : 45057

Dest IP : 2.2.2.2

Dest Port : 45058

Waveform:



Pada test case tersebut, dapat dilihat bahwa output permit bernilai ‘0’ pada akhir pengecekan rule_index 2 (aturan ACL ketiga). Hal ini karena aturan tidak tertulis ACL, yaitu “deny ip any any” sebagai aturan terakhir. Hal ini berarti hasil pengujian sesuai karena input tidak match dengan ketiga aturan tertulis ACL.

5) Pengujian dengan Testbench

Selain pengujian menggunakan simulasi manual, digunakan pula testbench dengan testcase yang sama untuk memastikan pengujian yang sebelumnya sesuai. Berikut adalah output TCL Console setelah testbench dijalankan:

```
Note: Starting Testbench
Time: 0 ps  Iteration: 0  Process: /acl_system_tb/stim_proc  File: C:/RIyan/Tugas/Perancangan Sistem Digital dan Praktikum/Praktikum/Proteusz/project_39/project_3
Note: TEST 1: Applying packet that should match Rule 0. EXPECT: PERMIT
Time: 3 ns  Iteration: 0  Process: /acl_system_tb/stim_proc  File: C:/RIyan/Tugas/Perancangan Sistem Digital dan Praktikum/Praktikum/Proteusz/project_39/project_3
Note: TEST 1 PASSED: Packet was correctly PERMITTED.
Time: 145500 ps  Iteration: 0  Process: /acl_system_tb/stim_proc  File: C:/RIyan/Tugas/Perancangan Sistem Digital dan Praktikum/Praktikum/Proteusz/project_39/proj
Note: TEST 2: Applying packet that should be denied by Rule 1. EXPECT: DENY
Time: 145500 ps  Iteration: 0  Process: /acl_system_tb/stim_proc  File: C:/RIyan/Tugas/Perancangan Sistem Digital dan Praktikum/Praktikum/Proteusz/project_39/proj
Note: TEST 2 PASSED: Packet was correctly DENIED.
Time: 305500 ps  Iteration: 0  Process: /acl_system_tb/stim_proc  File: C:/RIyan/Tugas/Perancangan Sistem Digital dan Praktikum/Praktikum/Proteusz/project_39/proj
Note: TEST 3: Applying ICMP packet that should match Rule 2. EXPECT: PERMIT
Time: 305500 ps  Iteration: 0  Process: /acl_system_tb/stim_proc  File: C:/RIyan/Tugas/Perancangan Sistem Digital dan Praktikum/Praktikum/Proteusz/project_39/proj
Note: TEST 3 PASSED: ICMP packet was correctly PERMITTED.
Time: 485500 ps  Iteration: 0  Process: /acl_system_tb/stim_proc  File: C:/RIyan/Tugas/Perancangan Sistem Digital dan Praktikum/Praktikum/Proteusz/project_39/proj
Note: TEST 4: Applying packet that matches no rules. EXPECT: DENY
Time: 485500 ps  Iteration: 0  Process: /acl_system_tb/stim_proc  File: C:/RIyan/Tugas/Perancangan Sistem Digital dan Praktikum/Praktikum/Proteusz/project_39/proj
Note: TEST 4 PASSED: Packet was correctly DENIED by default.
Time: 706500 ps  Iteration: 0  Process: /acl_system_tb/stim_proc  File: C:/RIyan/Tugas/Perancangan Sistem Digital dan Praktikum/Praktikum/Proteusz/project_39/proj
Note: All test cases finished.
```

Dari hasil tersebut, dapat dilihat bahwa output dari seluruh testcase sesuai, sehingga pengujian dapat dikatakan valid.

4. MASALAH IMPLEMENTASI

a. Sinkronisasi Timing

Karena adanya delay antara waktu fetching microinstruction dari microinstr_gen, fsm_interpreter tidak berjalan sebagaimana mestinya. Hal ini membuat saya menambahkan control signal “control” yang menahan state untuk tidak berpindah ke DECODE sebelum control bernilai 1.

Selain itu, sinkronisasi lebih sulit dilakukan karena arsitektur sistem yang 3 tingkat. Karena ACE Processor bekerja dengan cara JUMP ke instruksi terakhir jika tidak ada match antara input dan ACE, waktu pengecekan masing-masing ACE menjadi tidak menentu. Hal ini tidak bisa diselesaikan dengan control, sehingga saya menambahkan 1 control signal baru, yaitu “start” yang digunakan untuk memulai pemrosesan microinstruction pada fsm_interpreter.

b. Perubahan Implementasi

Awalnya, saya hanya menerapkan standard ACL yang hanya memfilter IP address sumber. Setelah beberapa pertimbangan, saya memutuskan untuk menerapkan extended ACL yang jauh lebih kompleks. Karena hal ini, dibutuhkan perombakan total untuk field microinstruction dan microinstr_gen itu sendiri.

Di samping itu, karena extended ACL mempunyai parameter yang jauh lebih banyak, dibutuhkan proses parsing dan pengecekan yang jauh lebih kompleks. Hal ini dilakukan dengan menambahkan address instruksi pada microinstr_gen dari yang awalnya 9 menjadi 27, dan menambahkan 1 instruksi baru, yaitu CMP_PR untuk mengakomodir field tambahan yang diperlukan.

5. LINK GITHUB

https://github.com/r-ediewitsch/PacketFilter

REFERENSI

- “Access-Lists (ACL),” GeeksforGeeks, https://www.geeksforgeeks.org/access-lists-acl/ (accessed Aug. 1, 2025).
- “Basic concepts and fundamentals of ACLs,” ComputerNetworkingNotes, https://www.computernetworkingnotes.com/ccna-study-guide/basic-concepts-andfundamentals-of-acls.html (accessed Aug. 1, 2025).
- Rayhan, “Network Address Translation & Access Control List,” LEARN.NETLAB, https://learn.netlabdte.com/docs/DMJ/NAT%20&%20ACL (accessed Aug. 1, 2025).