Lyra: A Cross-Platform Language and Compiler for Data Plane Programming on Heterogeneous ASICs

Jiaqi Gao, Ennan Zhai, Hongqiang Harry Liu, Rui Miao, Yu Zhou
Bingchuan Tian, Chen Sun, Dennis Cai, Ming Zhang, Minlan Yu
Programmable switches gain significant traction
Programmable switches gain significant traction
Data plane programming is still at an early stage ...
Data plane programming is still at an early stage …

Chip specific languages ~ Assembly languages
Running example: A network sequencer
Running example: A network sequencer

Linearizing the transactions for consensus protocols such as Paxos
Running example: A network sequencer

Linearizing the transactions for consensus protocols such as Paxos
Running example: A network sequencer

Linearizing the transactions for consensus protocols such as Paxos

Add sequence header to selected flows and drop others
Running example: A network sequencer

Linearizing the transactions for consensus protocols such as Paxos

Add sequence header to selected flows and drop others
Running example: A network sequencer

Linearizing the transactions for consensus protocols such as Paxos

Add sequence header to selected flows and drop others
Problem 1: Portability

Low level, chip-specific languages
Problem 1: Portability

Low level, chip-specific languages
Problem 1: Portability
Low level, chip-specific languages

```c
// P4_14
action a_shift_switch_id(server_id) {
  shift_left(sequence.seq, server_id, 28);
}
table shift_switch_id {
  reads {
    ipv4.src_ip: exact;
  }
  actions {
    a_shift_switch_id;
  }
}
action a_and_timestamp() {
  bit_and(ig_ts, ig_ts, 0x0FFFFFFF);
}
table and_timestamp {
  actions {
    a_and_timestamp;
  }
  default_action: a_and_timestamp();
}
action a_sequence_header() {
  add_header(sequence);
  bit_and(sequence.seq, sequence.seq, ig_ts);
}
table add_sequence_header {
  actions {
    a_sequence_header;
  }
  default_action: a_sequence_header();
}
```
Problem 1: Portability

Low level, chip-specific languages

```
// P4_14
action a_shift_switch_id(server_id) {
    shift_left(sequence.seq, server_id, 28);
}
table shift_switch_id {
    reads {
        ipv4.src_ip: exact;
    }
    actions {
        a_shift_switch_id;
    }
}

action a_and_timestamp() {
    bit_and(ig_ts, ig_ts, 0x0FFFFFFF);
}
table and_timestamp {
    actions {
        a_and_timestamp;
    }
    default_action: a_and_timestamp();
}

action a_sequence_header() {
    add_header(sequence);
    sequence.seq = (server_id << 28) & (ig_ts & 0x0FFFFFFF);
    bit_and(ig_ts, ig_ts, 0x0FFFFFFF);
}
table add_sequence_header {
    actions {
        a_sequence_header;
    }
    default_action: a_sequence_header();
}
```
Problem 1: Portability

Low level, chip-specific languages

Chip Vendors

Add ID  Add Timestamp

add_header(sequence);
sequence.seq = (server_id << 28) & (ig_ts & 0xFFFFFFFF);
bit_and(ig_ts, ig_ts, 0xFFFFFFFF);
}
table and_timestamp {
    actions {
        a_and_timestamp;
    }
    default_action: a_and_timestamp();
}
action a_sequence_header() {
    add_header(sequence);
    bit_and(sequence.seq, sequence.seq, ig_ts);
}
table add_sequence_header {
    actions {
        a_sequence_header;
    }
    default_action: a_sequence_header();
}
Problem 1: Portability

Low level, chip-specific languages

Chip Vendors

Languages

Add ID
Add Timestamp

Chip Vendors

Languages

Add ID
Add Timestamp

add_header(sequence);
sequence.seq = (server_id << 28) & (ig_ts & 0x0FFFFFFF);
bit_and(ig_ts, ig_ts, 0x0FFFFFFF);
}
table and_timestamp {
    actions {
        a_and_timestamp;
    }
    default_action: a_and_timestamp();
}
action a_sequence_header() {
    add_header(sequence);
    bit_and(sequence.seq, sequence.seq, ig_ts);
}
table add_sequence_header {
    actions {
        a_sequence_header;
    }
    default_action: a_sequence_header();
}
Problem 2: Extensibility

Cannot program across switches
Problem 2: Extensibility

Cannot program across switches
Problem 2: Extensibility

Cannot program across switches

Aggregation
1000 entries

Top of Rack
1000 entries

Filter table
Problem 2: Extensibility

Cannot program across switches

Aggregation
1000 entries

Top of Rack
1000 entries

Filter table
Problem 2: Extensibility

Cannot program across switches

- Aggregation
  1000 entries

- Top of Rack
  1000 entries

- Filter table
  300 entries
Problem 2: Extensibility

Cannot program across switches

Aggregation
1000 entries

Top of Rack
1000 entries

Filter table
300 entries
Problem 2: Extensibility

Cannot program across switches

Aggregation
1000 entries

Top of Rack
1000 entries

Filter table

300 entries

800 entries
Problem 2: Extensibility

Cannot program across switches

Aggregation
1000 entries

Top of Rack
1000 entries

Filter table
300 entries

- None
- Filter (300)
  - Routing (500)

- Filter (800)
  - Routing (500)

800 entries
Problem 2: Extensibility

Cannot program across switches

Aggregation
1000 entries

Top of Rack
1000 entries

Filter table
300 entries

Routing (500)

Filter (300)

Routing (500)

Routing (500)

1300 entries
Problem 2: Extensibility

Cannot program across switches

Aggregation
1000 entries

Top of Rack
1000 entries

Filter table

- None
  - filter (300)
    - routing (500)
    - 300 entries
  - filter (800)
    - routing (500)
    - 800 entries
  - filter (1000)
    - routing (500)
    - 1300 entries

Problem 2: Extensibility

Cannot program across switches
Problem 3: Composition

Co-locate with other programs
Problem 3: Composition

Co-locate with other programs

Add an ARP table with 300 entries to the ToR switch
Problem 3: Composition
Co-locate with other programs

Add an ARP table with 300 entries to the ToR switch

Aggregation
1000 entries

Top of Rack
1000 entries
Problem 3: Composition
Co-locate with other programs

Add an ARP table with 300 entries to the ToR switch

Aggregation
1000 entries

Top of Rack
1000 entries
Add an ARP table with 300 entries to the ToR switch

Aggregation
1000 entries

Top of Rack
1000 entries

Problem 3: Composition
Co-locate with other programs
Data plane programming is still at an early stage

**Portability**
- Migrate program across switches
  - Different languages
  - Different architectures
  - Different ASIC models

**Extensibility**
- Distribute program across multiple switches
  - Different roles (In-band network telemetry)
  - Expand memory or computation (Sequencer)

**Composition**
- Fit multiple programs into one switch
  - Function overlapping
Data plane programming is still at an early stage

**Portability**
- Migrate program across switches
  - Different languages
  - Different architectures
  - Different ASIC models

**Extensibility**
- Distribute program across multiple switches
  - Different roles (In-band network telemetry)
  - Expand memory or computation (Sequencer)

**Composition**
- Fit multiple programs into one switch
  - Function overlapping

---

C language lets you get close to the machine, without getting tied up in the machine

— Dr. Brian Kernighan
Lyra: A high-level data plane language & compiler
Lyra: A high-level data plane language & compiler

Lyra program

One-big-pipeline model
Lyra: A high-level data plane language & compiler

Lyra program

One-big-pipeline model

Lyra compiler

NPL (Trident-4)
P4 (Tofino 32Q)
P4 (Tofino 64Q)
P4 (Silicon One)
Lyra: A high-level data plane language & compiler

Lyra program

One-big-pipeline model

Portability: Language synthesizer

Lyra compiler

NPL (Trident-4)
P4 (Tofino 32Q)
P4 (Tofino 64Q)
P4 (Silicon One)
Lyra: A high-level data plane language & compiler

Lyra program

One-big-pipeline model

Lyra compiler

Portability: Language synthesizer

Topology-aware code allocation

NPL (Trident-4)

P4 (Tofino 32Q)

P4 (Tofino 64Q)

P4 (Silicon One)
Lyra: A high-level data plane language & compiler

Lyra program
One-big-pipeline model

Lyra compiler

Portability: Language synthesizer
Topology-aware code allocation
Chip-specific constraint encoding

NPL (Trident-4)
P4 (Tofino 32Q)
P4 (Tofino 64Q)
P4 (Silicon One)
Lyra language: One-big-pipeline model
Lyra language: One-big-pipeline model
Lyra language: One-big-pipeline model

Per-switch model

One big switch

One-big-switch model
Lyra language: One-big-pipeline model

Per-switch model

One-big-pipeline model

One-big-switch model
Lyra language: One-big-pipeline model

Per-switch model

One-big-pipeline model

One-big-switch model
Lyra language: One-big-pipeline model

Per-switch model

One-big-pipeline model

One-big-switch model
Lyra language: One-big-pipeline model

Per-switch model

One-big-pipeline model

One-big-switch model
// Sequencer.lyra
header_type sequence_t{
    bit[32] seq;
}
parser_node parse_sequence{
    extract_fields(sequence);
}

pipeline[SEQ] {sequencer};

algorithm sequencer {
    add_sequence();
    routing();
}

func add_sequence() {
    if (ipv4.src_ip in filter) {
        add_header(sequence);
        sequence.seq = (filter[ipv4.src_ip] << 28) \
            & (ig_ts & 0xFFFFFFFF);
    } else {
        drop();
    }
}
// Sequencer.lyra
header_type sequence_t{
    bit[32] seq;
}
parser_node parse_sequence{
    extract_fields(sequence);
}

pipeline[SEQ] {sequencer};

algorithm sequencer {
    add_sequence();
    routing();
}

func add_sequence() {
    if (ipv4.src_ip in filter) {
        add_header(sequence);
        sequence.seq = (filter[ipv4.src_ip] << 28) \
                        & (ig_ts & 0xFFFFFFFF);
    } else {
        drop();
    }
}
Lyra language: Program

```lyra
// Sequencer.lyra
header_type sequence_t{
    bit[32] seq;
}
parser_node parse_sequence{
    extract_fields(sequence);
}

pipeline[SEQ] {sequencer};

algorithm sequencer {
    add_sequence();
    routing();
}

func add_sequence() {
    if (ipv4.src_ip in filter) {
        add_header(sequence);
        sequence.seq = (filter[ipv4.src_ip] << 28) \
            & (ig_ts & 0x0FFFFFFF);
    } else {
        drop();
    }
}
```

- Packet
  - Header
  - Parser
// Sequencer.lyra
header_type sequence_t{
  bit[32] seq;
}
parser_node parse_sequence{
  extract_fields(sequence);
}

pipeline[SEQ] {sequencer};

algorithm sequencer {
  add_sequence();
  routing();
}

func add_sequence() {
  if (ipv4.src_ip in filter) {
    add_header(sequence);
    sequence.seq = (filter[ipv4.src_ip] << 28) \n      & (ig_ts & 0xFFFFFFFF);
  } else {
    drop();
  }
}
// Sequencer.lyra
header_type sequence_t{
    bit[32] seq;
}
parser_node parse_sequence{
    extract_fields(sequence);
}

pipeline[SEQ] {sequencer};

algorithm sequencer {
    add_sequence();
    routing();
}

func add_sequence() {
    if (ipv4.src_ip in filter) {
        add_header(sequence);
        sequence.seq = (filter[ipv4.src_ip] << 28) \& (ig_ts & 0xFFFFFFFF);
    } else {
        drop();
    }
}
// Sequencer.lyra
header_type sequence_t{
    bit[32] seq;
}
parser_node parse_sequence{
    extract_fields(sequence);
}

pipeline[SEQ] {sequencer};

algorithm sequencer {
    add_sequence();
    routing();
}

func add_sequence() {
    if (ipv4.src_ip in filter) {
        add_header(sequence);
        sequence.seq = (filter[ipv4.src_ip] << 28) \ 
                        & (ig_ts & 0xFFFFFFFF);
    } else {
        drop();
    }
}
// Sequencer.lyra
header_type sequence_t{
  bit[32] seq;
}
parser_node parse_sequence{
  extract_fields(sequence);
}
pipeline[SEQ] {sequencer};
algorithm sequencer {
  add_sequence();
  routing();
}

func add_sequence() {
  if (ipv4.src_ip in filter) {
    add_header(sequence);
    sequence.seq = (filter[ipv4.src_ip] << 28) \n      & (ig_ts & 0x0FFFFFFF);
  } else {
    drop();
  }
}
// Sequencer.lyra
header_type sequence_t{
    bit[32] seq;
}
parser_node parse_sequence{
    extract_fields(sequence);
}
pipeline[SEQ] {sequencer};

algorithm sequencer {
    add_sequence();
    routing();
}

func add_sequence() {
    if (ipv4.src_ip in filter) {
        add_header(sequence);
        sequence.seq = (filter[ipv4.src_ip] << 28) \ 
                        & (ig_ts & 0x0FFFFFFF);
    }
    else {
        drop();
    }
}
Lyra: A high-level data plane language & compiler

Lyra program
One-big-pipeline model

Lyra compiler

Parser
Preprocessor
Analyzer
Frontend

Language synthesizer
Chip-specific constraint
Extensibility
Backend

NPL (Trident-4)
P4 (Tofino 32Q)
P4 (Tofino 64Q)
P4 (Silicon One)
Lyra: A high-level data plane language & compiler

Frontend
- Parser
- Preprocessor
- Analyzer

Backend
- Language synthesizer
- Chip-specific constraint
- Extensibility

Lyra program

One-big-pipeline model

Lyra compiler

NPL (Trident-4)
P4 (Tofino 32Q)
P4 (Tofino 64Q)
P4 (Silicon One)
Frontend
frontend

```cpp
extern list<bit[32] ip>[300] filter;
if (ipv4.src_ip in filter) {
    add_header(sequence);
    sequence.seq = (filter[ipv4.src_ip] << 28) \n        & (ig_ts & 0x0FFFFFFF);
} else {
    drop();
}
```

Lyra program
```c
extern list<bit[32] ip>[300] filter;
if (ipv4.src_ip in filter) {
    add_header(sequence);
    sequence.seq = (filter[ipv4.src_ip] << 28) \n        & (ig_ts & 0x0FFFFFFF);
} else {
    drop();
}
```

Lyra program

```
p1 = ipv4.src_ip in filter;
    add_header(sequence);
    tmp1 = filter[ipv4.src_ip] << 28;
    tmp2 = ig_ts & 0x0FFFFFFF;
    sequence.seq = tmp1 & tmp2;
p2 = !(ipv4.src_ip in filter);
    drop();
```

Intermediate representation
extern list<bit[32] ip>[300] filter;
if (ipv4.src_ip in filter) {
    add_header(sequence);
    sequence.seq = (filter[ipv4.src_ip] << 28) & (ig_ts & 0x0FFFFFFF);
}
else {
    drop();
}

Lyra program

p1 = ipv4.src_ip in filter;
add_header(sequence);
tmpl = filter[ipv4.src_ip] << 28;
tmp2 = ig_ts & 0x0FFFFFFF;
sequence.seq = tmpl & tmp2;
p2 = !(ipv4.src_ip in filter);
drop();

One big pipeline model

No complex statements

Intermediate representation
```c
extern list<bit[32] ip>[300] filter;
if (ipv4.src_ip in filter) {
    add_header(sequence);
    sequence.seq = (filter[ipv4.src_ip] << 28) \n        & (ig_ts & 0x0FFFFFFF);
} else {
    drop();
}
```

Lyra program

One big pipeline model

No complex statements
No branches

Intermediate representation
```c
extern list<bit[32] ip>[300] filter;
if (ipv4.src_ip in filter) {
    add_header(sequence);
    sequence.seq = (filter[ipv4.src_ip] << 28) & (ig_ts & 0x0FFFFFFF);
}
else {
    drop();
}
```

Lyra program

One big pipeline model

- No complex statements
- No branches
- Has statement dependencies
Lyra: A high-level data plane language & compiler

One-big-pipeline model

Lyra program

Lyra compiler

Frontend
- Parser
- Preprocessor
- Analyzer

Backend
- Language synthesizer
- Chip-specific constraint
- Extensibility

NPL (Trident-4)
P4 (Tofino 32Q)
P4 (Tofino 64Q)
P4 (Silicon One)
Lyra: A high-level data plane language & compiler

- Language synthesizer
- Chip-specific constraint
- Extensibility
- Backend
Lyra: A high-level data plane language & compiler

- Language synthesizer
- Chip-specific constraint
- Extensibility
- Backend
Lyra: A high-level data plane language & compiler

- Language synthesizer
- Chip-specific constraint
- Extensibility

Portability: Compile Lyra into structured low level languages
Lyra: A high-level data plane language & compiler

- **Language synthesizer**
- **Chip-specific constraint**
- **Extensibility**

**Portability** Compile Lyra into structured low level languages

**Composition** Checks whether the program can fit into the switch
Lyra: A high-level data plane language & compiler

- **Language synthesizer**
- **Chip-specific constraint**
- **Extensibility**

- **Portability**
  Compile Lyra into structured low level languages

- **Composition**
  Checks whether the program can fit into the switch

- **Extensibility**
  Correctly deploy the program in the scope
Lyra: A high-level data plane language & compiler

- **Language synthesizer**
- **Chip-specific constraint**
- **Extensibility**

**Portability** → Compile Lyra into structured low level languages

**Composition** → Checks whether the program can fit into the switch

**Extensibility** → Correctly deploy the program in the scope
Languages have different programming paradigms
Languages have different programming paradigms

P4
Languages have different programming paradigms

Table oriented programming
Languages have different programming paradigms

Table oriented programming
Languages have different programming paradigms

Table oriented programming

Procedural oriented programming
Languages have different programming paradigms

Table oriented programming

Procedural oriented programming
Languages have different programming paradigms

Table oriented programming

- table 1
- table 2
- table 3
- table 4

Procedural oriented programming

P4 Program
Languages have different programming paradigms

Table oriented programming

Procedural oriented programming

P4 Program

P4 Table
Languages have different programming paradigms

Table oriented programming

P4 Program

Procedural oriented programming

P4 Table

match

action 1

action 2

table 1

table 2

table 3

table 4
Languages have different programming paradigms

Table oriented programming

P4 Program

Procedural oriented programming

P4 Table
Languages have different programming paradigms

Table oriented programming:
- Table 1
- Table 2
- Table 3
- Table 4

Procedural oriented programming:
- Table 1
  - Action 1
  - Statement 1
  - Statement 2
- Parallel
- Action 2

P4 Program

P4 Table
Predicate block
Predicate block

Group of statements that has the same predicate and no dependency
Predicate block

Group of statements that has the same predicate and no dependency

```plaintext
p1 = ipv4.src_ip in filter;
    add_header(sequence);
    tmp1 = filter[ipv4.src_ip] << 28;
    tmp2 = ig_ts & 0x0FFFFFFF;
    sequence.seq = tmp1 & tmp2;

p2 = !(ipv4.src_ip in filter);
    drop();
```
Predicate block

Group of statements that has the same predicate and no dependency

```
p1 = ipv4.src_ip in filter;
    add_header(sequence);
    tmp1 = filter[ipv4.src_ip] << 28;
    tmp2 = ig_ts & 0x0FFFFFFF;
    sequence.seq = tmp1 & tmp2;

p2 = !(ipv4.src_ip in filter);
    drop();
```
Predicate block

Group of statements that has the same predicate and no dependency

```plaintext
p1 = ipv4.src_ip in filter;
    add_header(sequence);
    tmp1 = filter[ipv4.src_ip] << 28;
    tmp2 = ig_ts & 0x0FFFFFFF;
    sequence.seq = tmp1 & tmp2;

p2 = !(ipv4.src_ip in filter);
    drop();
```
Predicate block

Group of statements that has the same predicate and no dependency

\[
p_1 = \text{ipv4.src_ip in filter};
\text{add_header(sequence)};
\text{tmp1} = \text{filter[ipv4.src_ip]} \ll 28;
\text{tmp2} = \text{ig_ts} \& 0x0FFFFFFF;
\text{sequence.seq} = \text{tmp1} \& \text{tmp2};
\]

\[
p_2 = !\text{ipv4.src_ip in filter};
\text{drop}();
\]
Predicate block

Group of statements that has the same predicate and no dependency

\[
p1 = \text{ipv4.src_ip in filter};
\text{add_header(sequence);}
\text{tmp1 = filter[ipv4.src_ip] \ll 28;}
\text{tmp2 = ig_ts \& 0x0FFFFFFF;}
\text{sequence.seq = tmp1 \& tmp2;}
\]

\[
p2 = !(\text{ipv4.src_ip in filter});
\text{drop();}
\]
Predicate block

Group of statements that has the same predicate and no dependency

```plaintext
p1 = ipv4.src_ip in filter;
   add_header(sequence);
   tmp1 = filter[ipv4.src_ip] << 28;
   tmp2 = ig_ts & 0x0FFFFFFF;
   sequence.seq = tmp1 & tmp2;

p2 = !(ipv4.src_ip in filter);
   drop();
```

```plaintext
ipv4.src_ip in filter

add_header(sequence);
   tmp1 = filter[ipv4.src_ip] << 28;
   tmp2 = ig_ts & 0x0FFFFFFF;
```
Predicate block

Group of statements that has the same predicate and no dependency

```python
p1 = ipv4.src_ip in filter;
    add_header(sequence);
    tmp1 = filter[ipv4.src_ip] << 28;
    tmp2 = ig_ts & 0x0FFFFFFF;
    sequence.seq = tmp1 & tmp2;

p2 = !(ipv4.src_ip in filter);
    drop();
```

Predicate block

Group of statements that has the same predicate and no dependency

\[ p_1 = \text{ipv4.src_ip in filter}; \]
\[ \text{add_header(sequence)}; \]
\[ \text{tmp1} = \text{filter[ipv4.src_ip]} \ll 28; \]
\[ \text{tmp2} = \text{ig_ts} \& \text{0x0FFFFFFF}; \]
\[ \text{sequence.seq} = \text{tmp1} \& \text{tmp2}; \]

\[ p_2 = !(\text{ipv4.src_ip in filter}); \]
\[ \text{drop}(); \]
Predicate block

Group of statements that has the same predicate and no dependency

```cpp
p1 = ipv4.src_ip in filter;
    add_header(sequence);
    tmp1 = filter[ipv4.src_ip] << 28;
    tmp2 = ig_ts & 0x0FFFFFFF;
    sequence.seq = tmp1 & tmp2;

drop();
```

```cpp
p2 = !(ipv4.src_ip in filter);
    drop();
```

```cpp
ipv4.src_ip in filter
add_header(sequence);
    tmp1 = filter[ipv4.src_ip] << 28;
    tmp2 = ig_ts & 0x0FFFFFFF;
sequence.seq = tmp1 & tmp2;
```

```cpp
!(ipv4.src_ip in filter)
    drop();
```
Predicate block

Group of statements that has the same predicate and no dependency

\[ \text{p1} = \text{ipv4.src_ip in filter}; \]
\[ \text{add_header(sequence);} \]
\[ \text{tmp1} = \text{filter[ipv4.src_ip]} << 28; \]
\[ \text{tmp2} = \text{ig_ts & 0x0FFFFFFF}; \]
\[ \text{sequence.seq} = \text{tmp1} & \text{tmp2}; \]

\[ \text{p2} = \text{!(ipv4.src_ip in filter)}; \]
\[ \text{drop();} \]

\[ \text{ipv4.src_ip in filter} \]
\[ \text{add_header(sequence);} \]
\[ \text{tmp1} = \text{filter[ipv4.src_ip]} << 28; \]
\[ \text{tmp2} = \text{ig_ts & 0x0FFFFFFF}; \]
\[ \text{sequence.seq} = \text{tmp1} & \text{tmp2}; \]

\[ \text{Dependent} \]
\[ \text{ipv4.src_ip in filter} \]

\[ \text{!(ipv4.src_ip in filter)} \]
\[ \text{drop();} \]
Predicate block

Group of statements that has the same predicate and no dependency

\[
p_1 = \text{ipv4.src_ip in filter}; \\
\quad \text{add_header(sequence)}; \\
\quad \text{tmp1 = filter[ipv4.src_ip] \ll 28}; \\
\quad \text{tmp2 = ig_ts & 0x0FFFFFFF}; \\
\quad \text{sequence.seq = tmp1 & tmp2}; \\
\]

\[
p_2 = \neg(\text{ipv4.src_ip in filter}); \\
\quad \text{drop>();}
\]
Predicate block

Group of statements that has the same predicate and no dependency

```
ipv4.src_ip in filter

add_header(sequence);
tmp1 = filter[ipv4.src_ip] << 28;
tmp2 = ig_ts & 0x0FFFFFFF;
sequence.seq = tmp1 & tmp2;

!(ipv4.src_ip in filter)
drop();
```

Dependent

```
ipv4.src_ip in filter

sequence.seq = tmp1 & tmp2;
```
Predicate block

Group of statements that has the same predicate and no dependency

```
ipv4.src_ip in filter

add_header(sequence);
tmpl = filter[ipv4.src_ip] << 28;
tmp2 = ig_ts & 0x0FFFFFFF;

sequence.seq = tmpl & tmp2;

!(ipv4.src_ip in filter)

drop();
```
Predicate block

Group of statements that has the same predicate and no dependency

filter table 1

ipv4.src_ip in filter

add_header(sequence);
tmpl = filter[ipv4.src_ip] << 28;
tmp2 = ig_ts & 0xFFFFFFFF;

ipv4.src_ip in filter

sequence.seq = tmpl & tmp2;

!(ipv4.src_ip in filter)

drop();
Predicate block

Group of statements that has the same predicate and no dependency

```
ipv4.src_ip in filter
add_header(sequence);
tmp1 = filter[ipv4.src_ip] << 28;
tmp2 = ig_ts & 0xFFFFFFFF;
sequence.seq = tmp1 & tmp2;
!ipv4.src_ip in filter
drop();
```
Predicate block

Group of statements that has the same predicate and no dependency

```
filter table 1

match src_ip

action 1 (input: server_id)
add_header(sequence);
tmp1 = filter[ipv4.src_ip] << 28;
tmp2 = ig_ts & 0x0FFFFFFF;
drop();

!ipv4.src_ip in filter
```

Dependent
```ipv4.src_ip in filter
sequence.seq = tmp1 & tmp2;
```
Predicate block

Group of statements that has the same predicate and no dependency

```plaintext
match src_ip

filter table 1

action 1 (input: server_id)

add_header(sequence);
tmp1 = filter[ipv4.src_ip] << 28;
tmp2 = ig_ts & 0x0FFFFFFF;

Dependent

ipv4.src_ip in filter

sequence.seq = tmp1 & tmp2;

drop();

action 2
```
Predicate block

Group of statements that has the same predicate and no dependency

filter table 1

action 1 (input: server_id)
- `add_header(sequence);`
- `tmp1 = server_id << 28;`
- `tmp2 = ig_ts & 0x0FFFFFFF;`

Dependent

ipv4.src_ip in filter
- `sequence.seq = tmp1 & tmp2;`

match src_ip

action 2
- `drop();`
Predicate block

Group of statements that has the same predicate and no dependency

filter table 1

match src_ip

action 1 (input: server_id)

add_header(sequence);

tmp1 = servere_id << 28;

tmp2 = ig_ts & 0x0FFFFFFF;

action 2

drop();

filter table 2

action 1

sequence.seq = tmp1 & tmp2;
Predicate block

Group of statements that has the same predicate and no dependency

```cpp
ipv4.src_ip in filter

Predicate block

Group of statements that has the same predicate and no dependency

```

```
filter table 1

action 1 (input: server_id)
add_header(sequence);
tmp1 = servere_id << 28;
tmp2 = tig_ts & 0xFFFFFFFF;

match src_ip

action 2
drop();

filter table 2

action 1
sequence.seq = tmpl & tmp2;

```
Lyra: A high-level data plane language & compiler

- **Language synthesizer**
- **Chip-specific constraint**
- **Extensibility**

---

**Portability** I Compile Lyra into structured low level languages

**Composition** I Checks whether the program can fit into the switch

**Extensibility** I Correctly deploy the program in the scope
Lyra: A high-level data plane language & compiler

- **Language synthesizer**
- **Chip-specific constraint**
- **Extensibility**

**Portability** → Compile Lyra into structured low level languages

**Composition** → Checks whether the program can fit into the switch

**Extensibility** → Correctly deploy the program in the scope
Target-specific resource encoding: RMT

Filter table 1

match src_ip

action 1 (input: server_id)

add_header(sequence);

tmp1 = server_id << 28;

tmp2 = ig_ts & 0xFFFFFFFF;

action 2

drop();

Filter table 2

action 1

sequence.seq = tmp1 & tmp2;
Target-specific resource encoding: RMT

filter table 1

match src_ip

action 1 (input: server_id)
add_header(sequence);
tmp1 = server_id << 28;
tmp2 = ig_ts & 0x0FFFFFFF;

action 2
drop();

filter table 2

action 1
sequence.seq = tmp1 & tmp2;

S_{filter_1} S_{filter_2}
Target-specific resource encoding: RMT

```
tmp2 = ig_ts & 0x0FFFFFFF;
sequence.seq = tmp1 & tmp2;
add_header(sequence);
drop();
```

```
match src_ip

action 1 (input: server_id)
add_header(sequence);
tmp1 = server_id << 28;
tmp2 = ig_ts & 0x0FFFFFFF;
```

```
action 1
sequence.seq = tmp1 & tmp2;
```

```
action 2
drop();
```

\[ S_{filter_1} < S_{filter_2} \]
Target-specific resource encoding: RMT

\[
S_{\text{filter}_1} < S_{\text{filter}_2}
\]

\[
0 \leq S_{\text{filter}_1} \leq 31
\]
Target-specific resource encoding: RMT

\[ S_{filter_1} < S_{filter_2} \]

\[ 0 \leq S_{filter_1} \leq 31 \]

\[ 0 \leq S_{filter_2} \leq 31 \]
Target-specific resource encoding: RMT

\[
S_{\text{filter}_1} < S_{\text{filter}_2}
\]

\[
0 \leq S_{\text{filter}_1} \leq 31
\]

\[
0 \leq S_{\text{filter}_2} \leq 31
\]
Target-specific resource encoding: RMT

filter table 1

match src_ip

action 1 (input: server_id)
add_header(sequence);
tmp1 = server_id << 28;
tmp2 = ig_ts & 0x0FFFFFFF;

action 2
drop();

filter table 2

sequence.seq = tmp1 & tmp2;

S_{filter_1} < S_{filter_2}
0 \leq S_{filter_1} \leq 31
SMT solver

S_{filter_1} = 0
S_{filter_2} = 1

0 \leq S_{filter_2} \leq 31
Target-specific resource encoding: RMT

filter table 1
- match src_ip
- action 1 (input: server_id)
  - add_header(sequence);
  - tmp1 = server_id << 28;
  - tmp2 = ig_ts & 0xFFFFFFFF;
- action 2
  - drop();

filter table 2
- action 1
  - sequence.seq = tmp1 & tmp2;

SRAM memory
TCAM memory
Packet Header Vector
Table num per stage

$S_{filter_1} < S_{filter_2}$
$0 \leq S_{filter_1} \leq 31$
$0 \leq S_{filter_2} \leq 31$

SMT solver

$S_{filter_1} = 0$
$S_{filter_2} = 1$
Lyra: A high-level data plane language & compiler

- **Portability**: Compile Lyra into structured low level languages
- **Composition**: Checks whether the program can fit into the switch
- **Extensibility**: Correctly deploy the program in the scope
Lyra: A high-level data plane language & compiler

- **Language synthesizer**
- **Chip-specific constraint**
- **Extensibility**
- **Backend**

- **Portability** → Compile Lyra into structured low level languages
- **Composition** → Checks whether the program can fit into the switch
- **Extensibility** → Correctly deploy the program in the scope
Pass information between switches
Pass information between switches

Pack downstream switches' required data into packet header
Pass information between switches

Pack downstream switches' required data into packet header

Aggregation
1000 entries

Top of Rack
1000 entries

filter (1000)

filter (300)
routing (500)

1300 entries
Pass information between switches

Pack downstream switches' required data into packet header

Aggregation
1000 entries

src: 10.0.0.1

filter (1000)

Top of Rack
1000 entries

filter (300)

routing(500)

1300 entries
Pass information between switches

Pack downstream switches' required data into packet header

Aggregation
1000 entries

filter (1000)

src: 10.0.0.1

Top of Rack
1000 entries

filter (300)

src: 10.0.0.1

routing (500)

table_hit (1)

1300 entries
Pass information between switches

Pack downstream switches' required data into packet header

Aggregation
1000 entries

filter (1000)

src: 10.0.0.1

top of Rack
1000 entries

filter (300)

routing (500)

src: 10.0.0.1
table_hit (1)

src: 10.0.0.1

ID: 1
T: 23

1300 entries
Pass information between switches

Pack downstream switches' required data into packet header

No code is duplicated

Aggregation
1000 entries

filter (1000)

src: 10.0.0.1

Top of Rack
1000 entries

filter (300)

routing (500)

src: 10.0.0.1

table_hit (1)

src: 10.0.0.1

ID: 1

T: 23

1300 entries
Pass information between switches

Pack downstream switches' required data into packet header

No code is duplicated
No code is missing
Pass information between switches

Pack downstream switches' required data into packet header

No code is duplicated
No code is missing
No reordering
Lyra: A high-level data plane language & compiler

Lyra program

Parser
Preprocessor
Analyzer
Frontend

Language synthesizer
Chip-specific constraint
Extensibility
Backend

One-big-pipeline model

Lyra compiler

NPL (Trident-4)
P4 (Tofino 32Q)
P4 (Tofino 64Q)
P4 (Silicon One)
Evaluation
Demo for portability, extensibility and composition
## Lyra can reduce resource usage

<table>
<thead>
<tr>
<th>Program</th>
<th>P4\textsubscript{14}</th>
<th>Lyra</th>
<th>Synthesized P4\textsubscript{14}</th>
<th>Synthesized NPL</th>
<th>Longest Code Path</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LoC / Logic LoC</td>
<td>Tables</td>
<td>Actions</td>
<td>Registers</td>
<td>LoC / Logic LoC</td>
</tr>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>8</td>
<td>0</td>
<td>207/62</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>193/46</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>197/47</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
<td>6</td>
<td>194/97</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
<td>40</td>
<td>372/153</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
<td>2</td>
<td>177/73</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
<td>5</td>
<td>150/69</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>113/43</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>72/31</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
<td>0</td>
<td>4151/2563</td>
</tr>
</tbody>
</table>
## Lyra can reduce resource usage

<table>
<thead>
<tr>
<th>Program</th>
<th>P4\textsubscript{14} LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
<th>Registers</th>
<th>Lyra LoC / Logic LoC</th>
<th>Lyra Compile Time</th>
<th>Tables</th>
<th>Actions</th>
<th>Registers</th>
<th>Synthesized P4\textsubscript{14}</th>
<th>Lyra Compile Time</th>
<th>Tables</th>
<th>Actions</th>
<th>Registers</th>
<th>Synthesized NPL</th>
<th>Lyra Compile Time</th>
<th>Tables</th>
<th>Registers</th>
<th>Longest Code Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>8</td>
<td>0</td>
<td>207/62</td>
<td>0.987s</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>0.78s</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>193/46</td>
<td>0.914s</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0.72s</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>197/47</td>
<td>0.897s</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0.73s</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
<td>6</td>
<td>194/97</td>
<td>1.352s</td>
<td>16</td>
<td>20</td>
<td>6</td>
<td>0.95s</td>
<td>9</td>
<td>6</td>
<td>6</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
<td>40</td>
<td>372/153</td>
<td>1.909s</td>
<td>12</td>
<td>14</td>
<td>40</td>
<td>1.17s</td>
<td>3</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
<td>2</td>
<td>177/73</td>
<td>1.530s</td>
<td>13</td>
<td>16</td>
<td>2</td>
<td>0.85s</td>
<td>6</td>
<td>2</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
<td>5</td>
<td>150/69</td>
<td>1.158s</td>
<td>6</td>
<td>11</td>
<td>5</td>
<td>0.84s</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>113/43</td>
<td>0.91s</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>0.70s</td>
<td>4</td>
<td>2</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>72/31</td>
<td>0.852s</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0.67s</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
<td>0</td>
<td>4151/2563</td>
<td>33.6s</td>
<td>131</td>
<td>363</td>
<td>0</td>
<td>19.4s</td>
<td>125</td>
<td>0</td>
<td>53</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lyra can reduce resource usage

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
<th>Registers</th>
<th>Lyra LoC / Logic LoC</th>
<th>P4_{14} Compile Time</th>
<th>Actions</th>
<th>Registers</th>
<th>Synthesized P4_{14} Compile Time</th>
<th>Actions</th>
<th>Registers</th>
<th>Longest Code Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>8</td>
<td>0</td>
<td>207/62</td>
<td>0.987s</td>
<td>8</td>
<td>7</td>
<td>0.78s</td>
<td>4</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>193/46</td>
<td>0.914s</td>
<td>5</td>
<td>5</td>
<td>0.72s</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>197/47</td>
<td>0.897s</td>
<td>6</td>
<td>6</td>
<td>0.73s</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
<td>6</td>
<td>194/97</td>
<td>1.352s</td>
<td>16</td>
<td>20</td>
<td>0.95s</td>
<td>9</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
<td>40</td>
<td>372/153</td>
<td>1.909s</td>
<td>12</td>
<td>14</td>
<td>1.17s</td>
<td>3</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
<td>2</td>
<td>177/73</td>
<td>1.530s</td>
<td>13</td>
<td>16</td>
<td>0.85s</td>
<td>6</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
<td>5</td>
<td>150/69</td>
<td>1.158s</td>
<td>6</td>
<td>11</td>
<td>0.84s</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>113/43</td>
<td>0.91s</td>
<td>8</td>
<td>7</td>
<td>0.70s</td>
<td>4</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>72/31</td>
<td>0.852s</td>
<td>4</td>
<td>4</td>
<td>0.67s</td>
<td>3</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
<td>0</td>
<td>4151/2563</td>
<td>33.6s</td>
<td>131</td>
<td>363</td>
<td>0.94s</td>
<td>125</td>
<td>0</td>
<td>53</td>
</tr>
</tbody>
</table>

compute

Lyra can reduce resource usage

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
<th>Registers</th>
<th>Lyra LoC / Logic LoC</th>
<th>Synthesized P4_{14} Compile Time</th>
<th>Actions</th>
<th>Registers</th>
<th>Synthesized NPL Compile Time</th>
<th>Actions</th>
<th>Registers</th>
<th>Longest Code Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>8</td>
<td>0</td>
<td>207/62</td>
<td>0.987s</td>
<td>8</td>
<td>7</td>
<td>0.78s</td>
<td>4</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>193/46</td>
<td>0.914s</td>
<td>5</td>
<td>5</td>
<td>0.72s</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>197/47</td>
<td>0.897s</td>
<td>6</td>
<td>6</td>
<td>0.73s</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
<td>6</td>
<td>194/97</td>
<td>1.352s</td>
<td>16</td>
<td>20</td>
<td>0.95s</td>
<td>9</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
<td>40</td>
<td>372/153</td>
<td>1.909s</td>
<td>12</td>
<td>14</td>
<td>1.17s</td>
<td>3</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
<td>2</td>
<td>177/73</td>
<td>1.530s</td>
<td>13</td>
<td>16</td>
<td>0.85s</td>
<td>6</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
<td>5</td>
<td>150/69</td>
<td>1.158s</td>
<td>6</td>
<td>11</td>
<td>0.84s</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>113/43</td>
<td>0.91s</td>
<td>8</td>
<td>7</td>
<td>0.70s</td>
<td>4</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>72/31</td>
<td>0.852s</td>
<td>4</td>
<td>4</td>
<td>0.67s</td>
<td>3</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
<td>0</td>
<td>4151/2563</td>
<td>33.6s</td>
<td>131</td>
<td>363</td>
<td>0.94s</td>
<td>125</td>
<td>0</td>
<td>53</td>
</tr>
</tbody>
</table>
Lyra can reduce resource usage

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
<th>Registers</th>
<th>LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
<th>Registers</th>
<th>Compile Time</th>
<th>Tables</th>
<th>Actions</th>
<th>Registers</th>
<th>Compile Time</th>
<th>Tables</th>
<th>Registers</th>
<th>Longest Code Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>8</td>
<td>0</td>
<td>207/62</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>0.987s</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0.78s</td>
<td>4</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>193/46</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0.914s</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0.72s</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>197/47</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0.897s</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0.73s</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
<td>6</td>
<td>194/97</td>
<td>16</td>
<td>20</td>
<td>6</td>
<td>1.352s</td>
<td>9</td>
<td>6</td>
<td>18</td>
<td>0.95s</td>
<td>9</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
<td>40</td>
<td>372/153</td>
<td>12</td>
<td>14</td>
<td>40</td>
<td>1.909s</td>
<td>3</td>
<td>40</td>
<td>20</td>
<td>1.17s</td>
<td>3</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
<td>2</td>
<td>177/73</td>
<td>13</td>
<td>16</td>
<td>2</td>
<td>1.530s</td>
<td>6</td>
<td>2</td>
<td>18</td>
<td>0.85s</td>
<td>6</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
<td>5</td>
<td>150/69</td>
<td>6</td>
<td>11</td>
<td>5</td>
<td>1.158s</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>0.84s</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>113/43</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>0.91s</td>
<td>4</td>
<td>2</td>
<td>12</td>
<td>0.70s</td>
<td>4</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>72/31</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0.852s</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>0.67s</td>
<td>3</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
<td>0</td>
<td>4151/2563</td>
<td>131</td>
<td>363</td>
<td>0</td>
<td>33.6s</td>
<td>125</td>
<td>0</td>
<td>53</td>
<td>19.4s</td>
<td>125</td>
<td>0</td>
<td>53</td>
</tr>
</tbody>
</table>
Lyra can reduce resource usage

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
<th>Registers</th>
<th>Lyra LoC / Logic LoC</th>
<th>Lyra Compile Time</th>
<th>Tables</th>
<th>Actions</th>
<th>Registers</th>
<th>Longest Code Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>P414</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lyra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synthesized P414</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synthesized NPL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
<td>6</td>
<td>194/97</td>
<td>1.352s</td>
<td>16</td>
<td>20</td>
<td>6</td>
<td>0.95s</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>197/47</td>
<td>0.897s</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0.73s</td>
</tr>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>8</td>
<td>0</td>
<td>207/62</td>
<td>0.987s</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>0.78s</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>193/46</td>
<td>0.914s</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0.72s</td>
</tr>
<tr>
<td>NetChain</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
<td>40</td>
<td>372/153</td>
<td>1.909s</td>
<td>12</td>
<td>14</td>
<td>40</td>
<td>1.17s</td>
</tr>
<tr>
<td>NetCache</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
<td>2</td>
<td>177/73</td>
<td>1.530s</td>
<td>13</td>
<td>16</td>
<td>2</td>
<td>0.85s</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
<td>5</td>
<td>150/69</td>
<td>1.158s</td>
<td>6</td>
<td>11</td>
<td>5</td>
<td>0.84s</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>113/43</td>
<td>0.91s</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>0.70s</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>72/31</td>
<td>0.852s</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0.67s</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
<td>0</td>
<td>4151/2563</td>
<td>33.6s</td>
<td>131</td>
<td>363</td>
<td>0</td>
<td>19.4s</td>
</tr>
</tbody>
</table>

Lyra can reduce resource usage.
Lyra can reduce resource usage

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
<th>Registers</th>
<th>Lyra LoC / Logic LoC</th>
<th>Lyra Compile Time</th>
<th>Synthesized P4_{14}</th>
<th>Synthesized NPL</th>
<th>Longest Code Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>8</td>
<td>0</td>
<td>207/62</td>
<td>0.987s</td>
<td>8</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>193/46</td>
<td>0.914s</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>197/47</td>
<td>0.897s</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
<td>6</td>
<td>194/97</td>
<td>1.352s</td>
<td>16</td>
<td>20</td>
<td>6</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
<td>40</td>
<td>372/153</td>
<td>1.909s</td>
<td>12</td>
<td>14</td>
<td>40</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
<td>2</td>
<td>177/73</td>
<td>1.530s</td>
<td>13</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
<td>5</td>
<td>150/69</td>
<td>1.158s</td>
<td>6</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>113/43</td>
<td>0.91s</td>
<td>8</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>72/31</td>
<td>0.852s</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
<td>0</td>
<td>4151/2563</td>
<td>33.6s</td>
<td>131</td>
<td>363</td>
<td>0</td>
</tr>
</tbody>
</table>

Lyra can reduce resource usage significantly compared to traditional P4_{14} and NPL implementations.
Lyra can reduce resource usage

<table>
<thead>
<tr>
<th>Program</th>
<th>P4_{14}</th>
<th>Lyra</th>
<th>Synthesized P4_{14}</th>
<th>Synthesized NPL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LoC / Logic LoC</td>
<td>Tables</td>
<td>Actions</td>
<td>Registers</td>
</tr>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
<td>6</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
<td>40</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
<td>0</td>
</tr>
</tbody>
</table>
Lyra can reduce resource usage

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
<th>Ingress INT</th>
<th>Transit INT</th>
<th>Egress INT</th>
<th>Speedlight</th>
<th>NetCache</th>
<th>NetChain</th>
<th>NetPaxos</th>
<th>flowlet_switching</th>
<th>simple_router</th>
<th>switch</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>6</td>
<td>7</td>
<td>21</td>
<td>96</td>
<td>16</td>
<td>11</td>
<td>8</td>
<td>4</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
<td>16</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>363</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>194/97</td>
<td>177/73</td>
<td>194/97</td>
<td>372/153</td>
<td>177/73</td>
<td>150/69</td>
<td>113/43</td>
<td>72/31</td>
<td>4151/2563</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.352s</td>
<td>1.530s</td>
<td>1.352s</td>
<td>1.909s</td>
<td>1.530s</td>
<td>1.158s</td>
<td>0.91s</td>
<td>0.852s</td>
<td>33.6s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td>13</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>16</td>
<td>11</td>
<td>7</td>
<td>4</td>
<td>363</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
<td>16</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>19.4s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.17s</td>
<td>1.85s</td>
<td>0.84s</td>
<td>0.70s</td>
<td>0.67s</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>53</td>
</tr>
</tbody>
</table>

Lyra

<table>
<thead>
<tr>
<th>Traffic</th>
<th>Compile Time</th>
<th>Tables</th>
<th>Registers</th>
<th>Longest Code Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress INT</td>
<td>0.78s</td>
<td>4</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Transit INT</td>
<td>0.72s</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Egress INT</td>
<td>0.73s</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Speedlight</td>
<td>0.95s</td>
<td>9</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>NetCache</td>
<td>1.17s</td>
<td>3</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>NetChain</td>
<td>0.85s</td>
<td>6</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>0.84s</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>0.70s</td>
<td>4</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>simple_router</td>
<td>0.67s</td>
<td>3</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>switch</td>
<td>19.4s</td>
<td>125</td>
<td>0</td>
<td>53</td>
</tr>
</tbody>
</table>

**NetCache**
- nc_hdr.op == NC_READ_REQUEST
- Table: check_cache_valid
  - check_cache_valid_act
- nc_hdr.op == NC_UPDATE_REPLY
- Table: set_cache_valid
  - set_cache_valid_act
Lyra can reduce resource usage

### NetCache

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
<th>Logic LoC</th>
<th>Longest Code Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>8</td>
<td>0.987s</td>
<td>8</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>7</td>
<td>0.72s</td>
<td>4</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>8</td>
<td>0.73s</td>
<td>4</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
<td>0.95s</td>
<td>9</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
<td>1.17s</td>
<td>3</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
<td>0.85s</td>
<td>6</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
<td>0.84s</td>
<td>3</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
<td>0.70s</td>
<td>4</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
<td>0.67s</td>
<td>3</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
<td>19.4s</td>
<td>125</td>
</tr>
</tbody>
</table>

**Lyra LoC**

- `nc_hdr.op == NC_READ_REQUEST`
  - Table: `check_cache_valid`
  - `check_cache_valid_act`
- `nc_hdr.op == NC_UPDATE_REPLY`
  - Table: `set_cache_valid`
  - `set_cache_valid_act`

**Lyra Compile Time**

- 0.73s
- 0.72s
- 0.72s
- 0.95s
- 1.17s
- 0.85s
- 0.84s
- 0.70s
- 0.67s
- 19.4s
Conclusion

- Lyra is the first high-level data plane language and compiler that achieves portability, extensibility and composition.

- Lyra offers a one-big-pipeline programming model and can generate runnable chip-specific code across multiple switches.

- The programs generated by Lyra use fewer hardware resources than human-written programs.
Thanks !