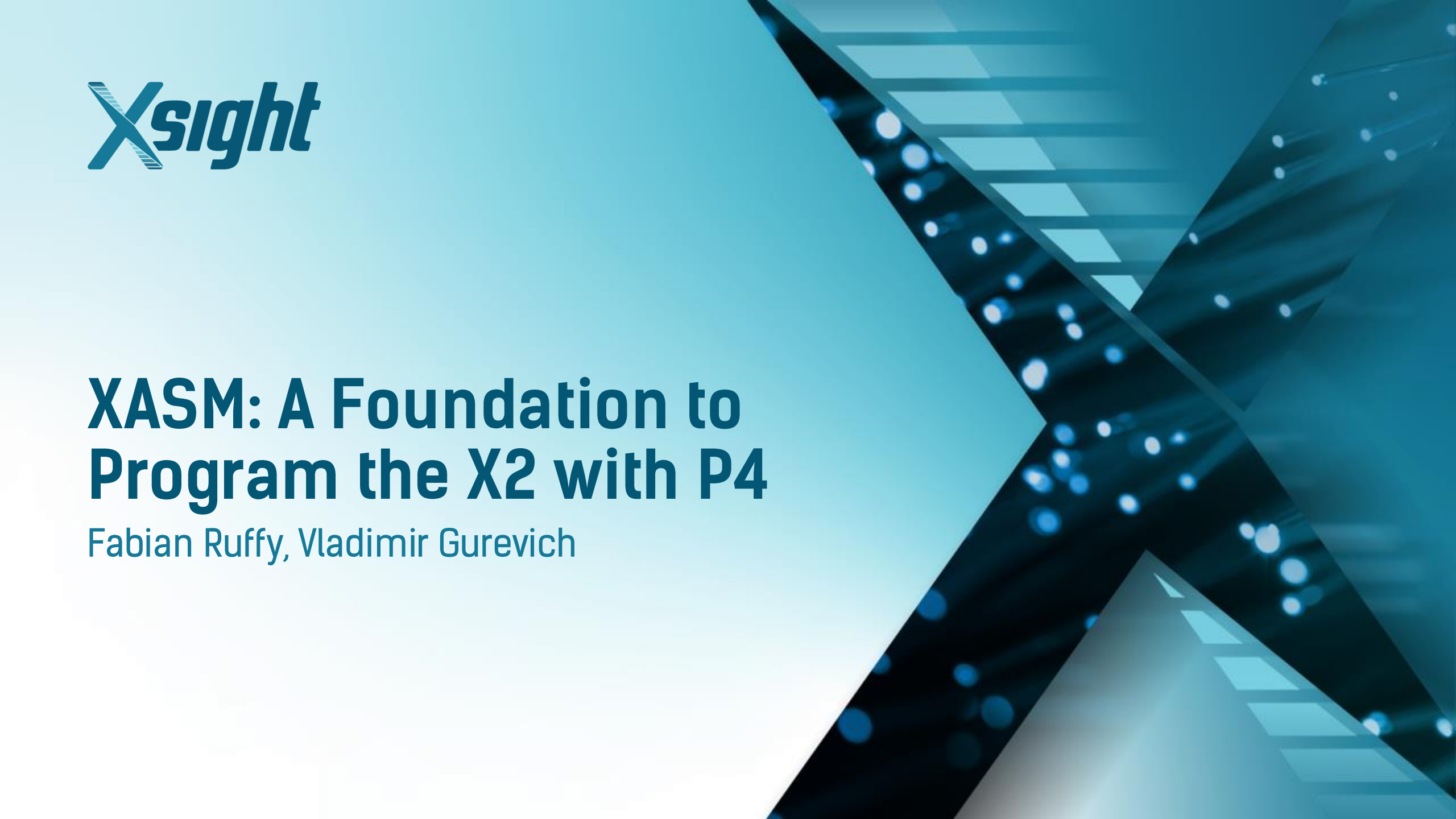


XASM: A Foundation to Program the X2 with P4

Fabian Ruffy, Vladimir Gurevich

Agenda

What is X2 and XISA?

Why XASM?

XASM Concepts

To P4 and Beyond!

Copyright© 2025 Xsight Labs. Public Presentation

The X2 Switch

12.8Gbps Programmable Switch

- 400Gbps/200Gbps/100Gbps ports
- Additional PCIe and Ethernet CPU ports

Ensemble of Data Processing Units (DPUs)

- Programmable Parser
- Programmable Match-Action Processor (MAP)
- Packet buffer, SRAMs, TCAMs

Open Instruction Set Architecture (XISA)

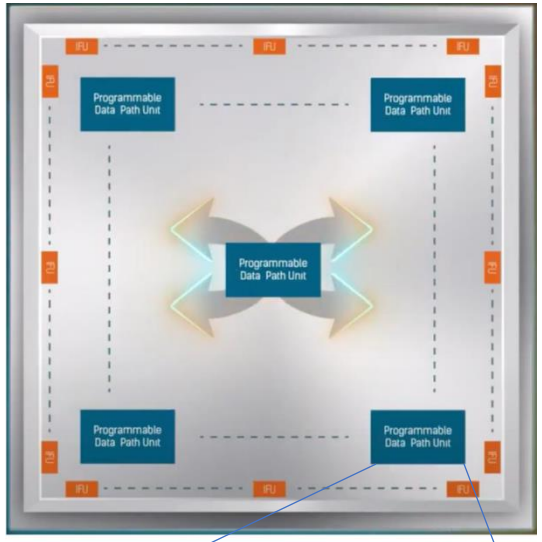


FIGURE 1. X-SWITCH SCALABLE ARCHITECTURE
(Source: Xsight Labs)

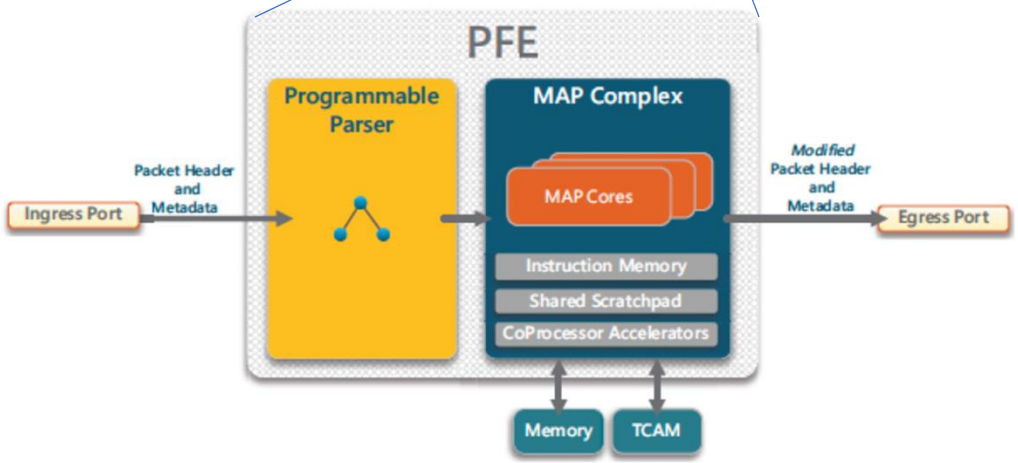
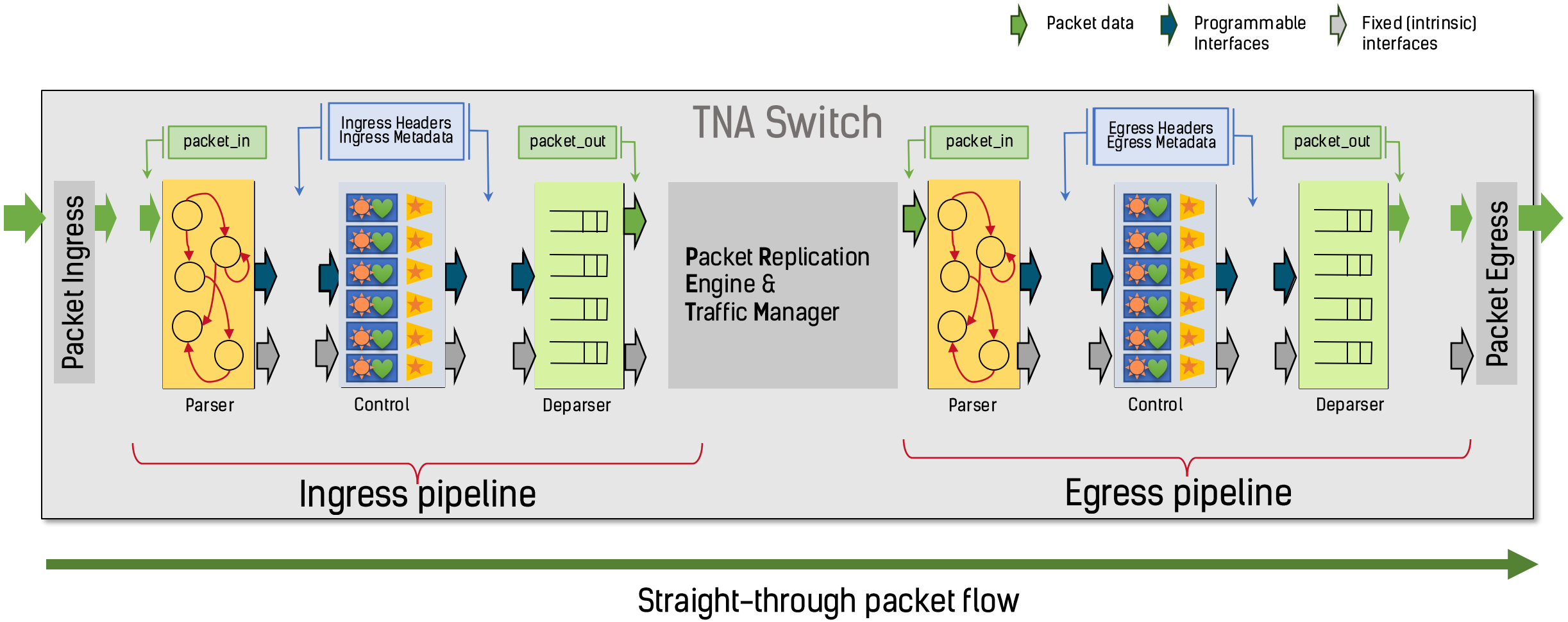


FIGURE 2. PROGRAMMABLE FORWARDING ENGINE
(Source: Xsight Labs)

Source: https://xsightlabs.com/wp-content/uploads/2025/03/WN-X-Switch-ISA-WP-Mar2025-1.pdf

Copyright© 2025 Xsight Labs. Public Presentation

Tofino Native Architecture (TNA)



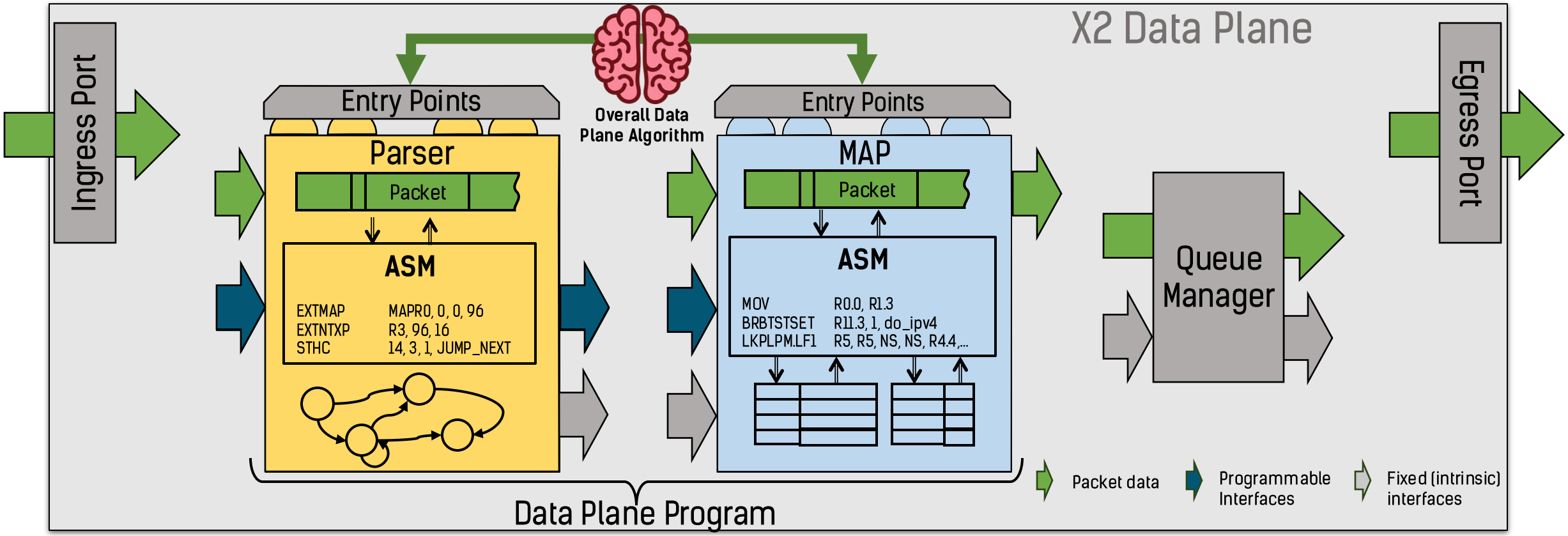
Copyright© 2025 Xsight Labs. Public Presentation

X2 Programming Model

Follows P4 Model: Fixed-function core with programmable portions

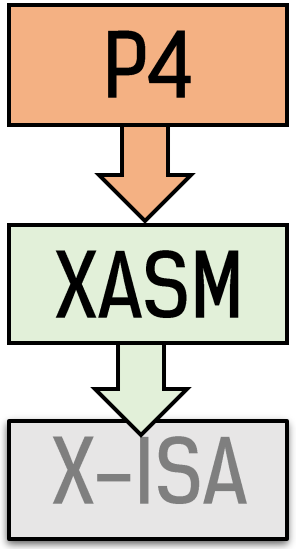
No fixed compute budget, throughput and latency degrades commensurate with complexity.

Programmable packet flow



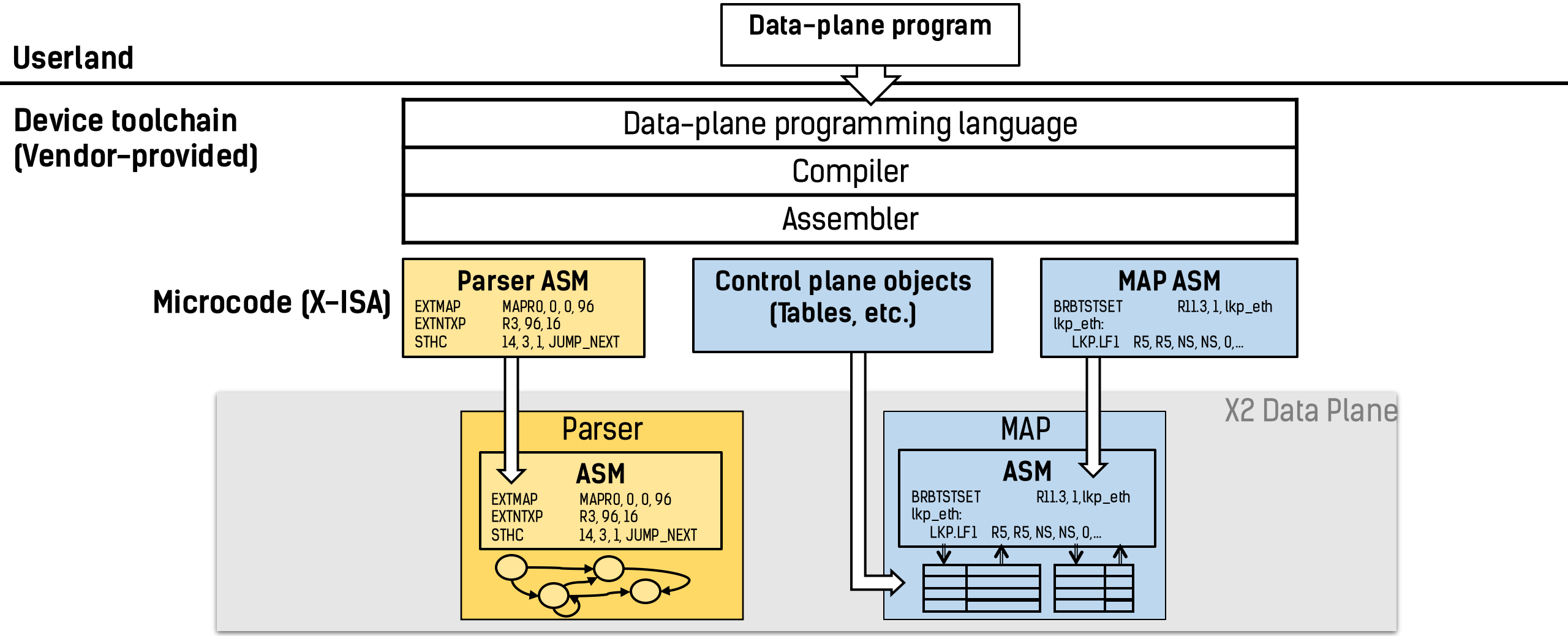
Copyright© 2025 Xsight Labs. Public Presentation

An Open ISA and XASM



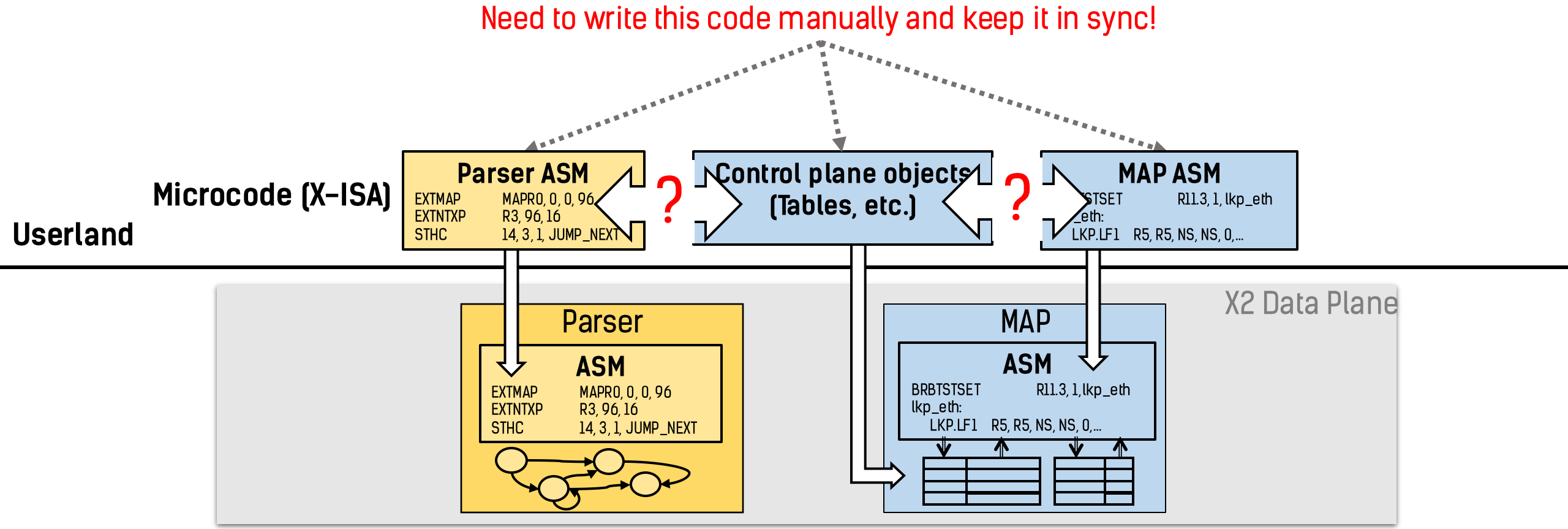
Confidential | Copyright© 2025 Xsight Labs. All Rights Reserved.

Compiling A Data-Plane Program



Copyright© 2025 Xsight Labs. Public Presentation

Down The Switch Stack With An Open ISA



Copyright© 2025 Xsight Labs. Public Presentation

Raw X-ISA Program

X2 Parser Instructions

```
entry_point_ethernet:
  EXTNXTP R0, 96, 16
  {// Transition Rules
    entry_point_ethernet
    0x800: entry_point_ipv4, 0, 0
  }
  STHC 14, 12, 1, 1
  HALT
entry_point_ipv4:
  EXTMAP MAPR4, 0, 0, 128
  EXTMAP MAPR2, 0, 128, 32
  STHC   20, 14, 3, 0
  HALT
entry_point_trap:
entry_point_loopback:
entry_point_reparse:
entry_point_fwc:
  HALTDROP
```

X2 MAP Instructions

```
ingress:
  BRBTSTCLR      R11.3, 14, not_ipv4_packet
  CONCAT.CD      R2.2, 0, R1.3, 0, 16
  LKPLPM.LF5.R   R1.3, R1.3, -1, -1, R2, R2, 0, 4
  SYNC.N         32, ipv4_lpm_miss
  AQMEG.LF3      R1.0, R1.3
  MOVI           R1.2, 0
  MOVI.CD        R0.3, 0
  SYNC           8
  BRBTSTSET      R1.0, 0, aqm_drop
  SENDOUT.H      R1, R0, 0

not_ipv4_packet:
ipv4_lpm_miss:
aqm_drop:
    SYNCALL      255
    DROP.H       0
```

Copyright© 2025 Xsight Labs. Public Presentation

Source: https://xsightlabs.com/wp-content/uploads/2025/04/XISA_Simple_IPv4.pdf

Requirements For Our XDE

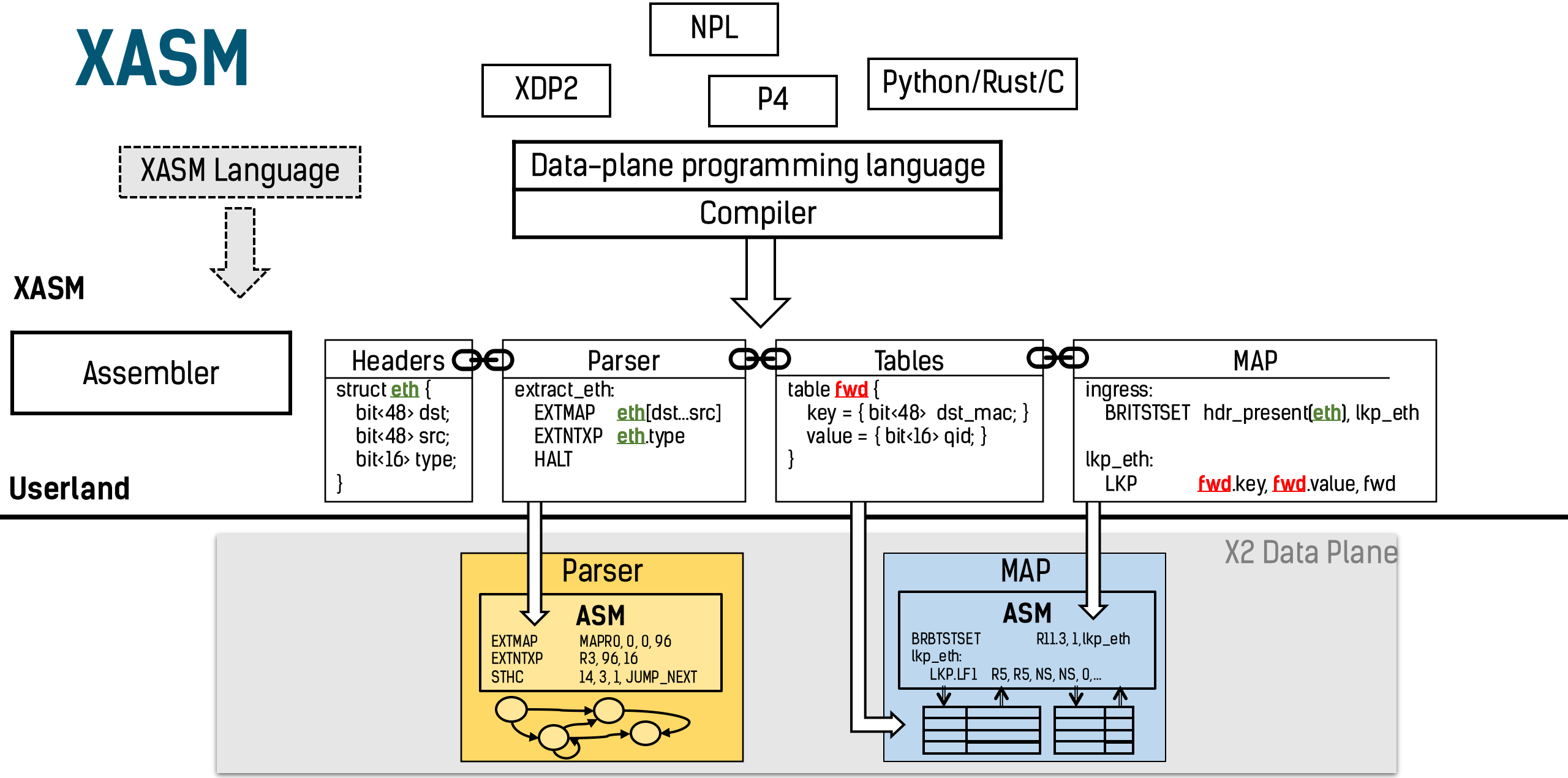
1. Make it **easy** to write data-plane programs and **hard** to make mistakes.

2. Give the same control/performance as X-ISA.

3. Export the right interfaces to high-level languages.

Copyright© 2025 Xsight Labs. Public Presentation

XASM



Copyright© 2025 Xsight Labs. Public Presentation

XASM Is Inspired By P4

Symbolic programming: All definitions are in the same place and work together.
Control plane interface is an explicit part of the program.

P4

Constants and Types

```
typedef bit<48> mac_addr_t;

header ethernet_h {
  mac_addr_t   dst;
  mac_addr_t   src;
  bit<16>      etype;
}
```

Tables and Externs

```
action send(port_id_t port) {…}
action drop() {…}
table ipv4_lpm {
  key = {
    meta.vrf     : exact;
    hdr.ipv4.dst : lpm;
  }
                 actions = { send; drop
```

Parser Code

```
state parse_ethernet {
  pkt.extract(hdr.eth);
  transition
  select(hdr.eth.etype) {
      0x0800 : parse_ipv4;
      default: accept;
    }
 }
```

Control Code

```
apply {
 if (hdr.ipv4.isValid()) {
   ipv4_lpm.apply();
 }
}
```

XASM

Constants and Types

```
typedef bit<48> mac_addr_t;

header ethernet_h {
    mac_addr_t   dst;
    mac_addr_t   src;
    bit<16>      etype;
}
```

Tables and Externs

```
table lpm ipv4_lpm {
  key = {
     bit<12>     vrf    : exact;
     ipv4_addr_t ipv4_dst : lpm;
  }
  value = {
                 bit<8>   sw_control;
                 bit<16>  qid;
                 bit<8>   action;
```

Parser Code

```
parse_ethernet:
 EXT          ether_r, ether_p
 NXTP         ether_r.ethertype
 MOVMAP       ethernet, ether_r
 STHC         ethernet_h,          \
              header_id_t.ETHERNET,\
              layer_id.L2,         \
              X2::NEXT_ON_MISS
 HALT
```

MAP Code

```
ingress:
BRBTSTCLR  hdr_valid[hdr_id.IPV4], \
           nonip
CONCAT     lpm_key.vrf, port_meta.vrf
LKPLPM.LF0 lpm_key, tbl_val, ipv4_lpm
LDSPI      lpm_val, ipv4_lpm_default
SYNC.N     [LF0], ipv4_lpm_default
MOV        lpm_val, tbl_val
nonip:     DROP.H 0
```

Copyright© 2025 Xsight Labs. Public Presentation

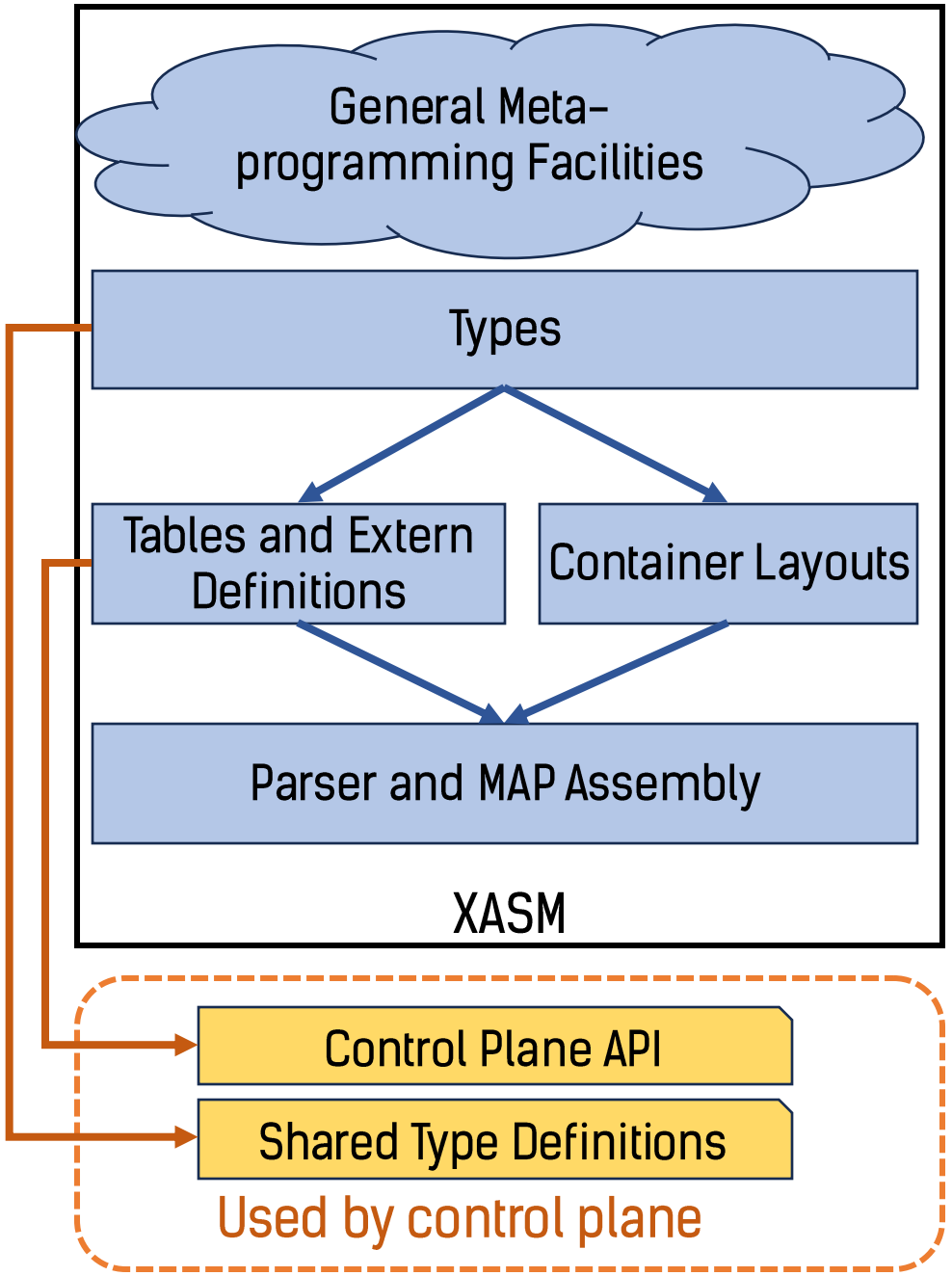
XASM: Summary

Macro-assembler with metaprogramming

- Functions, macros, and expressions.
- Custom errors, imports, conditionals.
- Definitions (types, tables, externs).

The goal for XASM definitions is to…

- …to make assembly code more readable.
- …to generate control plane APIs.
- …to combine all programmable X2 elements.



Copyright© 2025 Xsight Labs. Public Presentation

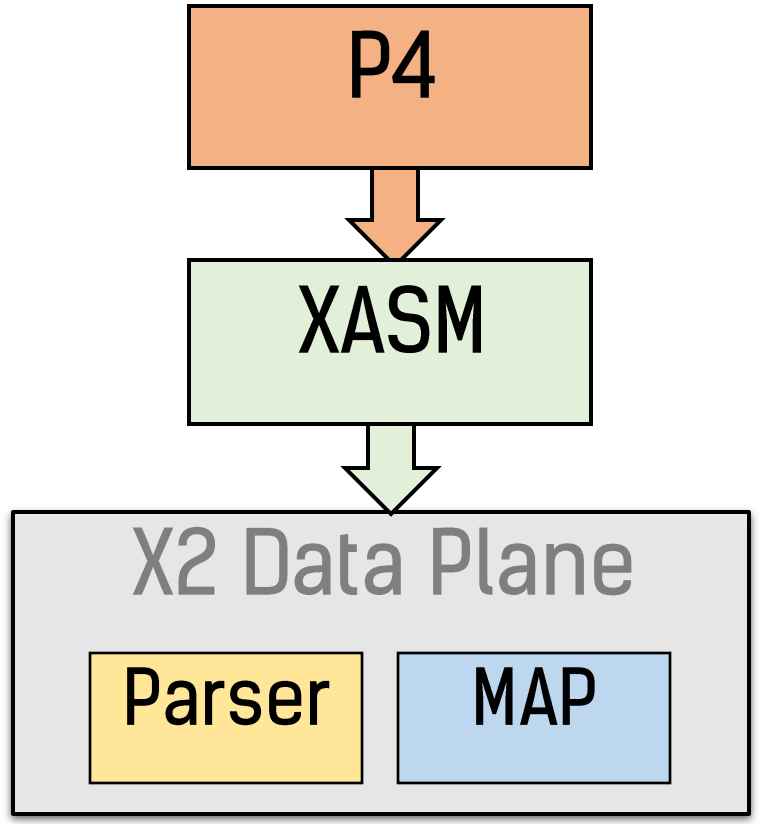
From XASM To P4

XASM is still assembly! Assembly is hard.

P4 is our language of choice to support higher-level data-plane programming!

We are building a P4 compiler!

- XASM is the foundation for this compiler.



Copyright© 2025 Xsight Labs. Public Presentation

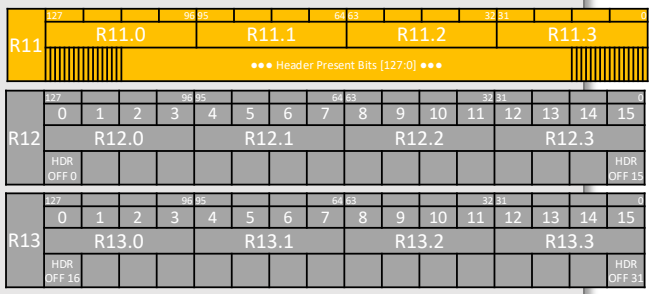
Header Definitions

P4

```
typedef bit<48> mac_addr_t;
typedef bit<32> ipv4_addr_t;

enum bit<16> ethertype_t {
    TPID = 0x8100,
    IPV4 = 0x0800
}

header ethernet_h {
    mac_addr_t   dst;
    mac_addr_t   src;
    ethertype_t  etype;
}

header ipv4_h {
    bit<4> ihl;
    bit<4> version;
    …
    ipv4_addr_t src;
    ipv4_addr_t dst;
}

header ipv6_h {
```

XASM

```
typedef bit<48> mac_addr_t;
typedef bit<32> ipv4_addr_t;

enum bit<16> ethertype_t {
    TPID = 0x8100,
    IPV4 = 0x0800
}

header ethernet_h {
    mac_addr_t   dst;
    mac_addr_t   src;
    ethertype_t  etype;
}

header ipv4_h {
    bit<4> ihl;
    bit<4> version;
    …
    ipv4_addr_t src;
    ipv4_addr_t dst;
}

enum bit<7> header_id_t { ETHERNET, IPV4, IPV6 };
enum bit<5> layer_id_t  { L2, L3 };
```



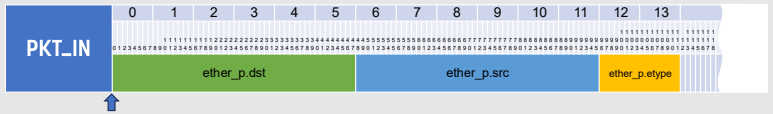
Copyright© 2025 Xsight Labs. Public Presentation

Extracting an Ethernet Header

P4

```
struct my_headers_t {
    ethernet_h ethernet;
}

parser MyParser(pkt_in pkt,
                out my_headers_t hdr, …) {

    state parse_ethernet {
        pkt.extract(hdr.ethernet);

        ethertype_t etype = hdr.ethernet.etype;

        transition select(etype) {
            ethertype_t.IPV4: parse_ipv4;
            default:          accept;
        }
    }

    state parse_ipv4 {
        pkt.extract(hdr.ipv4);
        transition accept;
}
```

XASM




```
parserdef MyParser {

assembly {

parse_ethernet:
    assume ethernet_h   ether_p   in PKT_IN;



    TransitionRules {
        ethertype_t.IPv4 -> parse_ipv4;
    }

    EXTMAP      ethernet[.dst:.src],  ether_p[.dst:.src]
    EXTNXTP.CD  ether,                ether_p.ethertype
    MOVMAP      ethernet.ethertype,   ethertype
    STHC        ether_h.size_bytes(), \
                header_id_t.ETHERNET, \
                layer_id.L2,          \
                X2::NEXT_ON_MISS
    HALT
```

Copyright© 2025 Xsight Labs. Public Presentation

Extracting an Ethernet Header

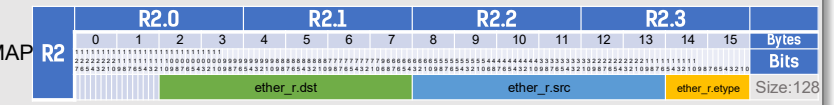
P4

```
struct my_headers_t {
    ethernet_h ethernet;
}

parser MyParser(pkt_in pkt,
                out my_headers_t hdr, …) {

    state parse_ethernet {
        pkt.extract(hdr.ethernet);

        ethertype_t etype = hdr.ethernet.etype;

        transition select(etype) {
            ethertype_t.IPV4: parse_ipv4;
            default:          accept;
        }
    }

    state parse_ipv4 {
        pkt.extract(hdr.ipv4);
        transition accept;
}
```

XASM

```
parserdef MyParser {

assembly {

parse_ethernet:
    assume ethernet_h   ether_p   in PKT_IN;
    assume ethernet_h   ethernet  in R2;



    TransitionRules {
        ethertype_t.IPv4 -> parse_ipv4;
    }

    EXTMAP      ethernet[.dst:.src],  ether_p[.dst:.src]
    EXTNXTP.CD  ether,                ether_p.ethertype
    MOVMAP      ethernet.ethertype,   ethertype
    STHC        ether_h.size_bytes(), \
                header_id_t.ETHERNET, \
                layer_id.L2,          \
                X2::NEXT_ON_MISS
    HALT
```

MAP R2



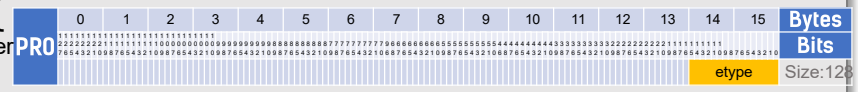
Copyright© 2025 Xsight Labs. Public Presentation

Extracting an Ethernet Header

P4

```
struct my_headers_t {
    ethernet_h ethernet;
}

parser MyParser(pkt_in pkt,
                out my_headers_t hdr, …) {

    state parse_ethernet {
        pkt.extract(hdr.ethernet);

        ethertype_t etype = hdr.ethernet.etype;

        transition select(etype) {
            ethertype_t.IPV4: parse_ipv4;
            default:          accept;
        }
    }

    state parse_ipv4 {
        pkt.extract(hdr.ipv4);
        transition accept;
}
```

XASM




```
parserdef MyParser {

assembly {

parse_ethernet:
    assume ethernet_h   ether_p   in PKT_IN;
    assume ethernet_h   ethernet  in R2;
    assume ethertype_t  etype     in PR0;

    TransitionRules {
        ethertype_t.IPv4 -> parse_ipv4;
    }

    EXTMAP      ethernet[.dst:.src],  ether_p[.dst:.src]
    EXTNXTP.CD  ether,                ether_p.ethertype
    MOVMAP      ethernet.ethertype,   ethertype
    STHC        ether_h.size_bytes(), \
                header_id_t.ETHERNET, \
                layer_id.L2,          \
                X2::NEXT_ON_MISS
    HALT
```

Copyright© 2025 Xsight Labs. Public Presentation

Extracting an Ethernet Header

P4

```
struct my_headers_t {
    ethernet_h ethernet;
}

parser MyParser(pkt_in pkt,
                out my_headers_t hdr, …) {

    state parse_ethernet {
        pkt.extract(hdr.ethernet);

        ethertype_t etype = hdr.ethernet.etype;

        transition select(etype) {
            ethertype_t.IPV4: parse_ipv4;
            default:          accept;
        }
    }

    state parse_ipv4 {
        pkt.extract(hdr.ipv4);
        transition accept;
}
```

XASM

```
parserdef MyParser {

assembly {

parse_ethernet:
    assume ethernet_h   ether_p    in PKT_IN;
    assume ethernet_h   ethernet   in R2;
    assume ethertype_t  etype      in PR0;

    TransitionRules {
        ethertype_t.IPv4 -> parse_ipv4;
    }

    EXTMAP      ethernet[.dst:.src],  ether_p[.dst:.src]
    EXTNXTP.CD  ether,                ether_p.ethertype
    MOVMAP      ethernet.ethertype,   ethertype
    STHC        ether_h.size_bytes(), \
                header_id_t.ETHERNET, \
                layer_id.L2,          \
                X2::NEXT_ON_MISS
    HALT
```

Copyright© 2025 Xsight Labs. Public Presentation

Defining a Match Table

P4

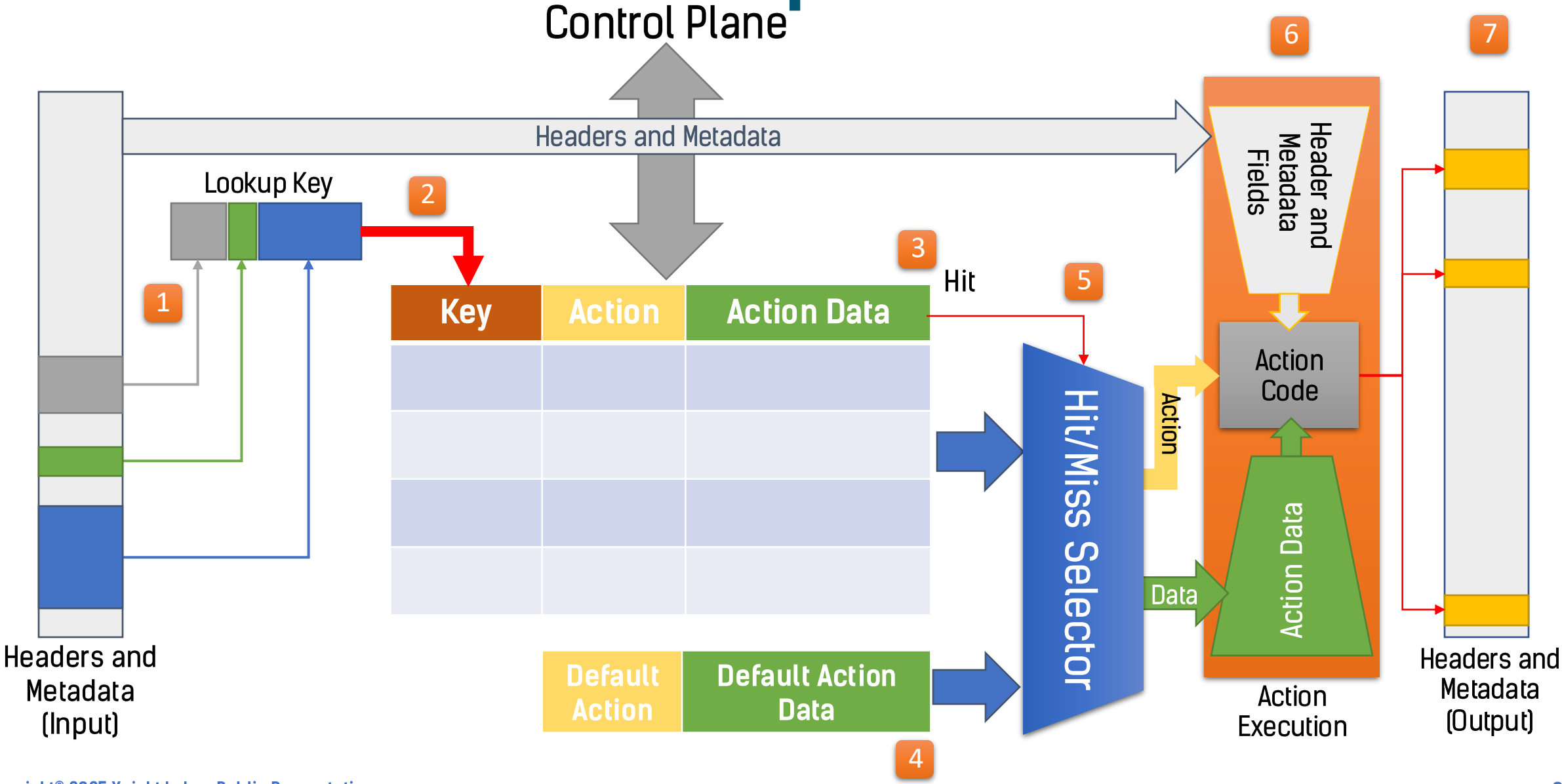
```
action send(queue_id_t qid) {
    if (X2_is_full(qid)) {
        X2_drop();
    } else {
        X2_simple_send(qid);
    }
}

action drop() {
    X2_drop();
}

table ipv4_lpm {
    key = {
        meta.vrf     : exact;
        hdr.ipv4.dst : lpm;
    }
    actions = {
        send; drop;
    }
    size = 65536;
}
```

XASM

```
table lpm ipv4_lpm {
    key = {
        bit<12>     vrf      : exact;
        ipv4_addr_t ipv4_dst : lpm;
    }
    value = {
        bit<1>   valid;
        bit<7>   _;      /* Padding */
        bit<16>  qid;
        bit<6>   _;      /* Padding */
        bit<2>   action; /* Bit encoded */
    }
    properties = {
        size         = 65536;
        duplicate    = X2::DUPLICATE_TO_ALL;
        mem_region   = "4x4";
        cache_enable = true;
    }
}
```

Copyright© 2025 Xsight Labs. Public Presentation

Match-Action Table Operation



Copyright© 2025 Xsight Labs. Public Presentation

Performing Match-Action Operation

P4

```
control MyIngress(inout my_headers_t hdr, …) {
    action send() { … }
    action drop() { … }
    table ipv4_lpm { … }

    apply {
        . . .
        if (hdr.ipv4.isValid()) {
            ipv4_lpm.apply();
        }
        ...
    }
}
```

XASM

```
mapdef MyMap {
    table  lpm ipv4_lpm { … }

Assembly {
    assume X2::header_valid_t  header_valid  in R11;
    assume ipv4_lpm::key       lpm_key       in R3.2..R3.3;
    assume ipv4_lpm::value     table_val     in R3.0;
    assume ipv4_lpm::value     lpm_val       in R3.1;

    BRBTSTCLR  hdr_valid[header_id_t.IPV4],  not_ipv4_packet
    CONCAT     lpm_key.vrf,  port_metadata.vrf                    1
    LKPLPM.LF0 lpm_key,  table_val, ipv4_lpm                      2
    LDSPI      lpm_val,  ipv4_lpm_default_entry                   4
    SYNC.N     [LF0],    ipv4_lpm_miss                            5
    MOV        lpm_val,  table_val      /* This is a hit */       3

ipv4_lpm_miss:
    JTL.NM     lpm_val.action,     \                              6
               ipv4_lpm_send, ipv4_lpm_drop, ipv4_lpm_done
ipv4_lpm_send:
    …
    BRI        ipv4_lpm_done

ipv4_lpm_drop:
    DROP       0
    BRI        ipv4_lpm_done
ipv4_lpm_done:                                                    7
    …

not_ipv4_packet:
```

Copyright© 2025 Xsight Labs. Public Presentation

The Opportunity With the X2: Choice

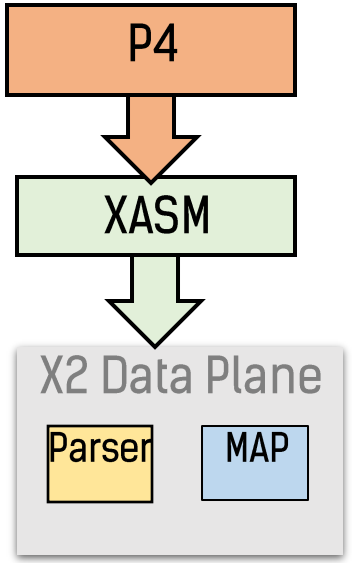
The open ISA gives you lots of flexibility on the X2 hardware!

Performance scales gracefully.

- Complex data plane code might be slower but always compiles.
- There are many ways to write very efficient code.

How far can we go with P4? We are not sure.

- We are certain about one approach, but there might be others!
- Many P4 architectures can be implemented.
- You can experiment with other languages.



Copyright© 2025 Xsight Labs. Public Presentation

Build X2 Applications With Us!

The X2 is a new programmable switch on the block.

- It has an open ISA!

We want you to experiment or build data-plane programs on the X2!

- We make this easy with XASM.
- We are working on P4 as the first high-level language to program the X2.

When is this software available? Stay tuned…

- Talk to us! fabianr@xsightlabs.com, vladimirg@xsightlabs.com

Copyright© 2025 Xsight Labs. Public Presentation