P4 Language Design Working Group

Gordon Brebner

XILINX®

Language Design Working Group

- **Responsibilities**
  - Defining the P4 language specification
  - Managing the graceful evolution of the language
- **Membership**
  - Co-chairs: Gordon Brebner and Nate Foster
  - Open to representatives of all members of P4.org
- **Activities**
  - Regular electronic discussion on `p4-design` email list
  - In-person design meetings at Stanford
- **Process**
  - Members propose new features and develop prototype implementations
  - Working group reviews proposals and updates the specification

Recent Updates

- **Finalized design of $P4_{16}$ – spec released yesterday on P4.org**
- **Target-architecture separation**

  Enables portability across platforms and extensibility through "externs"

- **Static type system**

  Offers rich constructs for structuring data and finding bugs early in development

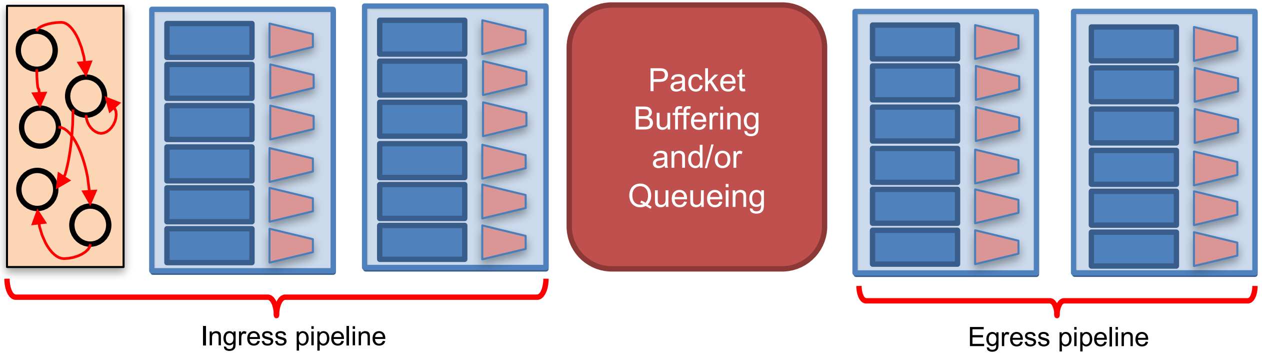
- **Higher-level programming constructs**

  Makes programs more succinct and encourages code reuse

- **Open-source prototype implementation available (`p4c` +`Bmv2`)**

Target-Architecture Separation

P4$_{14}$ is based on the PISA abstract forwarding model



Components

- Parser
- Ingress + egress controls

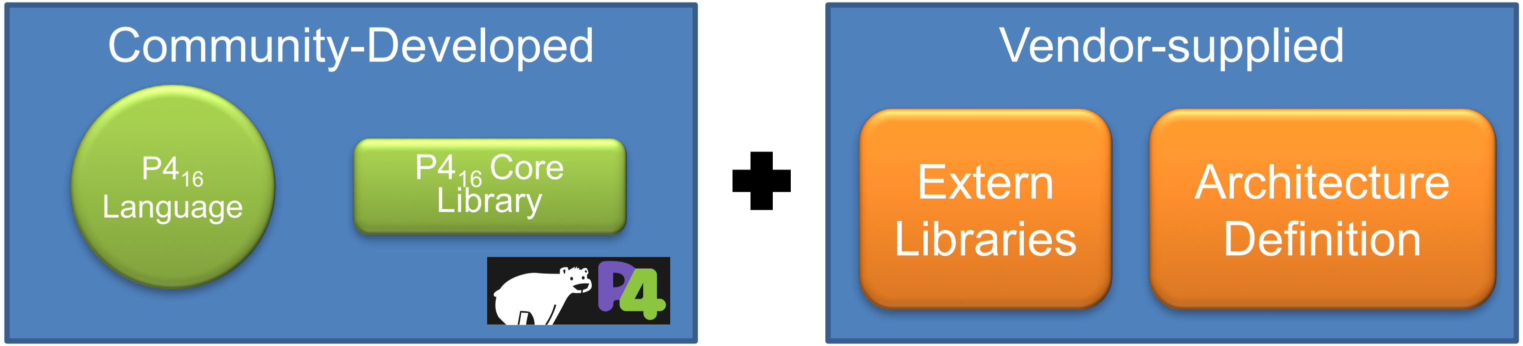
Limitations

- Architecture is not the natural and/or only fit for every target (e.g., FPGA)
- Difficult to extend the language with new functionality (e.g., checksums)

Target-Architecture Separation

P4$_{16}$ introduces the notion of an architecture model

- Collection of P4-programmable blocks
- Interfaces between blocks
- Available "extern" functions and stateful objects



Number of keywords reduced from > 70 to < 40 because of this

Example Architecture: v1model.p4

```
// Standard metadata carried between components
struct standard_metadata_t {
    bit<9>  ingress_port;
    bit<9>  egress_spec;
    ...
}
// Extern checksum object
extern Checksum16 {
    Checksum16();
    bit<16> get<D>(in D data);
}
// Programmable parser
parser Parser<H, M>(packet_in b,
                    out H parsedHdr,
                    inout M meta,
                    inout standard_metadata_t standard_metadata);
// Programmable ingress pipeline
control Ingress<H, M>(inout H hdr,
                      inout M meta,
                      inout standard_metadata_t standard_metadata);
...
// Top-level switch package
package V1Switch<H, M>(Parser<H, M> p,
                       VerifyChecksum<H, M> vr,
                       Ingress<H, M> ig,
                       Egress<H, M> eg,
                       ComputeChecksum<H, M> ck,
                       Deparser<H> dep);
```

- **Can define the PISA model used in P4$_{14}$ as an architecture instance for P4$_{16}$**
- **P4-programmable components:**
  - Parser
  - Checksum verification
  - Ingress pipeline
  - Egress pipeline
  - Checksum computation
  - Deparser
- **Extern objects:**
  - Counters
  - Meters
  - Registers
  - Checksum units
- **Programs are portable across *any* target that implements this architecture**
- **Compiler back-end maps P4 fragments into target-specific code**

Static Type System

- **Rich constructs for structuring data**
  - Headers, header stacks, header unions
  - Structs, enums, tuples, sets
- **Numeric types**
  - `bit<W>`: unsigned, fixed-width bit strings
  - `varbit<W>`: unsigned, variable-width bit strings
  - `int<W>`: signed, fixed-width integers
  - `int`: arbitrary precision integer constants

All numeric operations now have a well-defined semantics

Higher-level Programming Constructs

- **P4$_{16}$ supports a rich sub-language of expressions**

  `modify_field(egress_port,4)` ➔ `egress_port = 2+2`

- **P4$_{16}$ lifts many of the restrictions imposed in P4$_{14}$**
  - No longer need to define a dummy table to execute code in a control
  - Actions may contain non-trivial control-flow (e.g., conditionals)

- **Parameters and constructors facilitate code reuse**

```
control fwd(in ipv4_h ipv4)() {
  table t {
    key = {
      ipv4.dstAddress : ternary;
    }
    ...
  }
}
```

Next Steps

Architectures and APIs

- New architectures
- Control-plane interactions
- Incremental reprogramming

Tools and Implementation

- New targets
- Automated analysis and verification

Experience

- More innovative applications built on P4 – over to you …