Created By	**Hemant Singh**

< Nested Bit-vector Structure and Flatten Midend Pass Support in p4c >

Reviewers

Department	Name/Title
	Mihai Bidiu, p4-dev

Modification History

Revision	Date	Originator	Comments
1	08/31/2018	HS	First draft
1.1	09/04/2018	HS	Updated requirements
1.12	09/05/2018	HS	Updated doc for P4Runtime impact
1.121	09/05/2015	HS	Fixed two typos

Table of Contents

1 Purpose

1.1 Scope

The document covers this PR: https://github.com/p4lang/p4c/issues/562

A Use case is provided below:

The programmable switch asic we developed a p4c backed for, uses union of data structures. P4 only supports union of P4 headers via a header_union. Thus, our backend data structures are P4 headers. Further, the asic hardware sub-element (Mux Comparator) uses a data structure with bit-fields and structures. This use case demands structures in headers. A Mux Comparator, common to many switching asics, performs comparisons in parallel. The hardware has finite number of instructions to compare data. If a data structure is un-rolled, the Mux Comparator runs out of instructions. The Mux has enough width but not enough instructions to compare. This is why nested bit-vector structs work with this Mux. We used the simplest solution by changing p4c frontend to support nested bit-vector for struct and P4 header along with Cast support. The backend didn't have to change.

The nested structures or headers are expected as parameters to at least P4 Control and table key. Alternatives may exist to solve the problem, but may complicate the backend. A structure overlay creates more complexity in the backend vs. a nested bit-vector structure. Also, nested bit-vector struct requires Cast support as well. Do you plan to add Cast support with structure overlay.

Specifically, p4c is modified to support:

1. Nested bit-vector structures.
2. Nested bit-vector structures in P4 header.
3. Casting of nested bit-vector structure to bit-field of same width or vice versa.
4. Item 1 and 2 as parameters in P4 Action, Control, and Parser.
5. The bmv2 backend changes to support items 1 and 2.
6. A P4 header_union with headers including nested bit-vector structure(s).
7. Only P4-16.

A nested header is not supported.

Additionally, a table key or any other P4 type used by P4Runtime cannot use items 1 and 2 above. This rule allows control-plane API generation to not be impacted. P4Runtime supports Tables, PSA Action Profiles, Meters, Counters, Packet Replication, Parser Values, Registers, Digests, and Externs.

Another goal of the PR is to develop a midend pass to flatten any nested bit-vector structures. Specifically, p4c is modified to support:

1. Sweeping over the whole P4 program and flattening any global and local nested bit-vector structures.
2. Sweeping over the whole P4 program to change use of any bit-vector structures to flattened form. An example use is an assignment statement, or a table key. During use, an issue exists, see the example code below.
   `ip = (bit<32>) b.sip; // b is a bit-vector struct with "sip_t sip"` as a bit-vector struct for member. The struct sip has already been flattened before the above line of code is processed. Therefore, two assignment statements are generated with bit Slices used in LHS of the assignment statement.
3. If a parameter of a P4 Action, Control, and Parser is a nested bit-vector structure, the parameter flattened as a tuple akin to what is implemented in current `p4c/midend/nestedStructs.h[.cpp]`. Alternatively, the parameter could be left untouched (not flattened). Use of the parameter inside the Control block is always in flattened form.