

P4FPGA: High Level Synthesis for Networking

Han Wang, Ki Suh Lee, Vishal Shrivastav, Hakim Weatherspoon
Cornell University

1 Introduction

Field-programmable gate arrays (FPGAs) are often used to prototype custom packet processing algorithms. FPGAs are appealing because they balance the speed of hardware and the flexibility of software. From a programming language perspective, domain specific languages (DSLs), such as P4 and PX, are emerging to allow succinct expression of packet processing algorithms. What remains difficult, is the process to translate high level DSLs to low-level hardware description language, such as Verilog, in order to generate FPGA firmware. In this work, we present an open-source framework that enables compilation of high-level P4 language to FPGA firmware. Our framework translates a given P4 program through a number of intermediate steps. It leverages two existing intermediate representations of a P4 program, and generates corresponding Bluespec System Verilog (BSV) for simulation and synthesis. In addition, we provide runtime support for a packet processing algorithm to enable communication with network interfaces and host OS through PCI express bus. We have implemented a number of P4 applications on top of our framework to show its general applicability.

2 Design

P4FPGA framework automatically generates P4 pipeline implementation from a P4 program, and the control-plane API to configure the run-time aspect of a P4 pipeline. In addition, the framework provides infrastructure support to allow a generated P4 pipeline to communicate with other peripherals in a given platform, e.g. 10G ethernet interface and PCI express interconnect. Figure 1 shows an example of generated pipeline for P4 datapath.

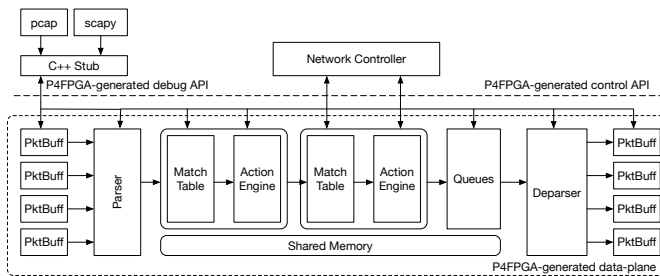


Figure 1: Generated P4 Pipeline

2.1 Runtime Support

Runtime implements the part of system that is outside of the scope of P4 specification. For example, P4 specification does not specify how packet memory are managed on FPGA. As a result, it is up to the runtime to implement a memory management unit (MMU), and provide an MMU interface to P4 pipeline. Furthermore, P4FPGA supports both Altera and Xilinx FPGAs, which means the runtime has to be able to support intellectual property (IP) cores provided in both Quartus and Vivado. The runtime is responsible to hide these details from a packet processing pipeline, and provide an abstract packet model to P4 pipeline. In our case, a packet is identified by a single 32-bit ID, which is used as a token throughout the pipeline.

2.2 Software Hardware Interface

Host communicates with data-plane through a set of auto-generated application programming interfaces (APIs). With these APIs, user have direct access to every state elements in data-plane at run-time. For example, a controller implemented as a user-space process can program a hardware match table to insert or modify flow rules. A test program can access hardware packet buffer to inject sample packets to verify the correctness of parser and deparser. P4FPGA also provides built-in library to enable replay of tcpdump trace to verify the end-to-end correctness of pipeline.

2.3 P4 Constructs

P4FPGA provides parameterized templates for P4-style match and action engine. We have implemented exact match table based on a fully-associative content addressable memory (CAM). P4FPGA can be extended to support third-party CAM implementations with better resource-efficiency and performance. Parser, deparser and control flow constructs are implemented as finite state machines (FSMs). Bluespec provides a built-in language construct to express FSM, which simplifies the design of P4FPGA framework.

3 Conclusion

P4FPGA automates the generation of P4-based packet processing pipeline to FPGA firmware. It is the first open-source tool to provide such capability. We have implemented P4FPGA and are currently in evaluation phase to obtain performance figure and resource utilization. In summary, P4FPGA greatly reduces the effort to design packet processing prototype in FPGA. With P4FPGA, we help to reduce the design time of a prototype from weeks to days.