**P4→NetFPGA: A low-cost solution for testing P4 programs in hardware**

Short Version

The P4→NetFPGA workflow provides a seamless path for developers, who may be unfamiliar with hardware description languages, to compile their P4 programs directly to the NetFPGA SUME board. The intention is to provide an affordable platform that allows anyone to easily run out their P4 programs in hardware at line rate.

Quick Links

* [P4->NetFPGA Documentation](https://github.com/NetFPGA/P4-NetFPGA-public/wiki)
* [Registration page for access to github repo (free)](https://goo.gl/forms/h7RbYmKZL7H4EaUf1)
* [Request P4→NetFPGA tools and licenses through Xilinx University Program](https://www.xilinx.com/support/university/donation-program.html)
* [Request NetFPGA SUME board for academic users](https://netfpga.wufoo.com/forms/netfpga-special-pricing-request/)
* [NetFPGA Contributor Agreement](https://github.com/NetFPGA/NetFPGA-SUME-public/wiki/Contributing-Code#contributing-for-the-first-time)
* [Download Xilinx P4-SDNet](http://xilinx.com/sdnet)

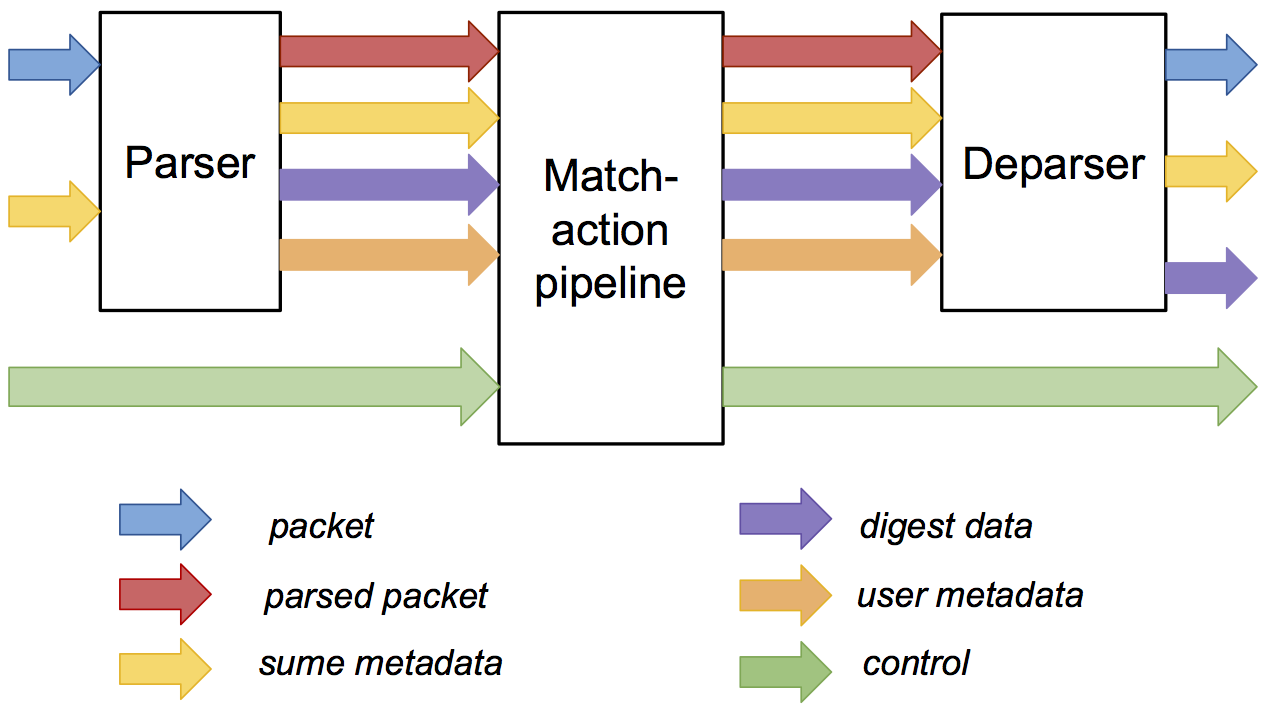
Long Version

P4 has emerged as an easy-to-use language for declaring how a forwarding device should process packets. A number of programmable targets are already available, ranging from programmable ASICs, software switches, and programmable NICs. This blog post is about a design flow we have developed (and are supporting) for compiling your P4 programs to the NetFPGA SUME platform - a low-cost platform widely used by universities for teaching and research. The [NetFPGA SUME board](http://store.digilentinc.com/netfpga-sume-virtex-7-fpga-development-board/) is manufactured and distributed by Digilent, and is normally available to academic users for less than $2000. (Users associated with a university should submit a request for academic pricing by filling out [this form](https://netfpga.wufoo.com/forms/netfpga-special-pricing-request/).)

The NetFPGA family of open-source platforms - designed for teaching and research - allow rapid prototyping of networking applications that run at line-rate in hardware. The latest NetFPGA platform, NetFPGA-SUME, has I/O capabilities for 100 Gbps operation, enabling researchers to prototype high-performance applications in hardware. While this platform is extremely powerful, it does require developers to be familiar with hardware description languages (e.g. Verilog or VHDL) and the FPGA development process, which is a steep learning curve for those without a background in hardware design. As a result, some researchers avoid prototyping with the NetFPGA platform and instructors find it difficult to use it as a tool with which to teach networking concepts. Our goal is to change this with the P4→NetFPGA workflow, making it much easier to try out new ideas in hardware.

The P4→NetFPGA workflow, which uses the Xilinx P4-SDNet tools, provides a seamless path for developers, who may be unfamiliar with hardware description languages, to compile their P4 programs directly to NetFPGA SUME. The intention is to provide an affordable platform that allows anyone to easily run out their P4 programs in hardware at line rate.

We take advantage of P416’s architecture specification mechanism to define the SimpleSumeSwitch architecture (shown below) for the NetFPGA platform. This architecture consists of a single parser, match-action pipeline, and deparser. It is ideal for new P4 developers to start experimenting with because, unlike the portable switch architecture (PSA), it is simple and easy to understand; and at the same time it is flexible enough to implement many networking protocols/algorithms. In contrast to the PSA, the SimpleSumeSwitch does not try to be fully comprehensive and include all features that would be needed by a commodity switch; its goal is to provide just enough features so that it is an effective prototyping tool. See [this link](https://github.com/NetFPGA/P4-NetFPGA-public/wiki/Workflow-Overview#simplesumeswitch-architecture) for more information about the SimpleSumeSwitch and a description of the different metadata buses in the architecture.



*SimpleSumeSwitch Architecture*

In an effort to abstract away HDL details from the P4 programmer, the P4→NetFPGA workflow provides a library of extern functions that can be called from P4 programs. These extern functions allow P4 programs to perform various atomic stateful operations, checksums, hash functions, and more. See [this link](https://github.com/NetFPGA/P4-NetFPGA-public/wiki/Workflow-Overview#p4-netfpga-extern-library) for a full list of the supported extern functions. The workflow also makes it very easy for users to add support for their own custom extern functions without having to modify any existing code. This should help to encourage developers to contribute new extern functions so that others may use them as well. While the tools require an HDL implementation of the extern functions, these functions may themselves be implemented using a high level synthesis tool that generates the HDL version.

The workflow generates both a C and Python API, built on top of SDNet runtime, to manipulate tables and read/write stateful memory on the switch. The P4 developer can choose to write their control-plane in either language. The Python API is intuitive and easy to use, allowing developers to quickly prototype designs, making use of the vast array of standard Python modules including [Python Scapy](https://scapy.readthedocs.io/en/latest/introduction.html). The API shares some of the same basic functionality as the recently proposed [P4Runtime](http://p4.org/api/announcing-p4runtime-a-contribution-by-the-p4-api-working-group/); and looking forward we plan to fully embrace the new proposal. In addition to the API, the workflow also generates an interactive CLI tool that allows the P4 developer to interact with the switch in real time as well as to query various compile time information about the switch.

There are a number of aspects about the toolchain that are not subject to public contribution; such as the P4 compiler itself and the underlying API functions that are provided to interact with tables. However, there is still many ways to get involved! The NetFPGA project is driven by open source contributions from people such as yourself; so we strongly you to contribute in any way you see fit. Some possible ways to contribute include:

* P4 projects that target the SimpleSumeSwitch architecture
* Extern function implementations
* Bug fixes or improvements for: reference projects, extern functions, scripts, or templates
* Performance analysis tools and benchmarks

P4→NetFPGA has already been used to implement a number of different applications:

* Ethernet learning switch
* IPv4 Router
* TCP/IP flow size monitor
* IP packet fuzzer
* In-band network telemetry (INT)
* Heavy-hitter detection
* ECN-enabled hardware switch
* New proactive congestion control techniques that require switch support

As well as some unconventional networking applications such as an AI to play tic-tac-toe.

Please see the quick links listed above for more information about the P4->NetFPGA workflow. We look forward to the exciting projects you will produce using these tools. For questions or comments, please send an email to [sibanez@stanford.edu](mailto:sibanez@stanford.edu).