Georgia Institute of Technology

Dr. Nick Feamster
Associate Professor

Software Defined Networking

In this course, you will learn about software defined networking and how it is changing the way communications networks are managed, maintained, and secured.

School of Computer Science

Module 5.3: Programmable Data Plane

- Two Lessons
  - Programming the data plane: Click
  - **Scaling programmable data planes**
    - Making software faster
    - **Making hardware more programmable**
- **Optional** programming assignment (in Click)
- Quiz on Concepts

What Do We Want From SDN?

- Protocol-independent processing
- Control and repurpose in the field
- Implemented by fast, low-power chips

Today: Still Real Hardware Constraints

- OpenFlow is protocol dependent because of constraints of traditional switching chips.
- Mapping to existing switching chips enabled quick adoption.
- What if we could re-design the data plane?

Insight: Few Data Plane Primitives

- The set of functions that we want to perform on packets are pretty limited
  - Bit shifting
  - Parsing and rewriting header fields
  - …
- **Can build a flexible data plane by developing modules + ways to integrate them.**

Two Examples of Modular Programmable Hardware Data Planes

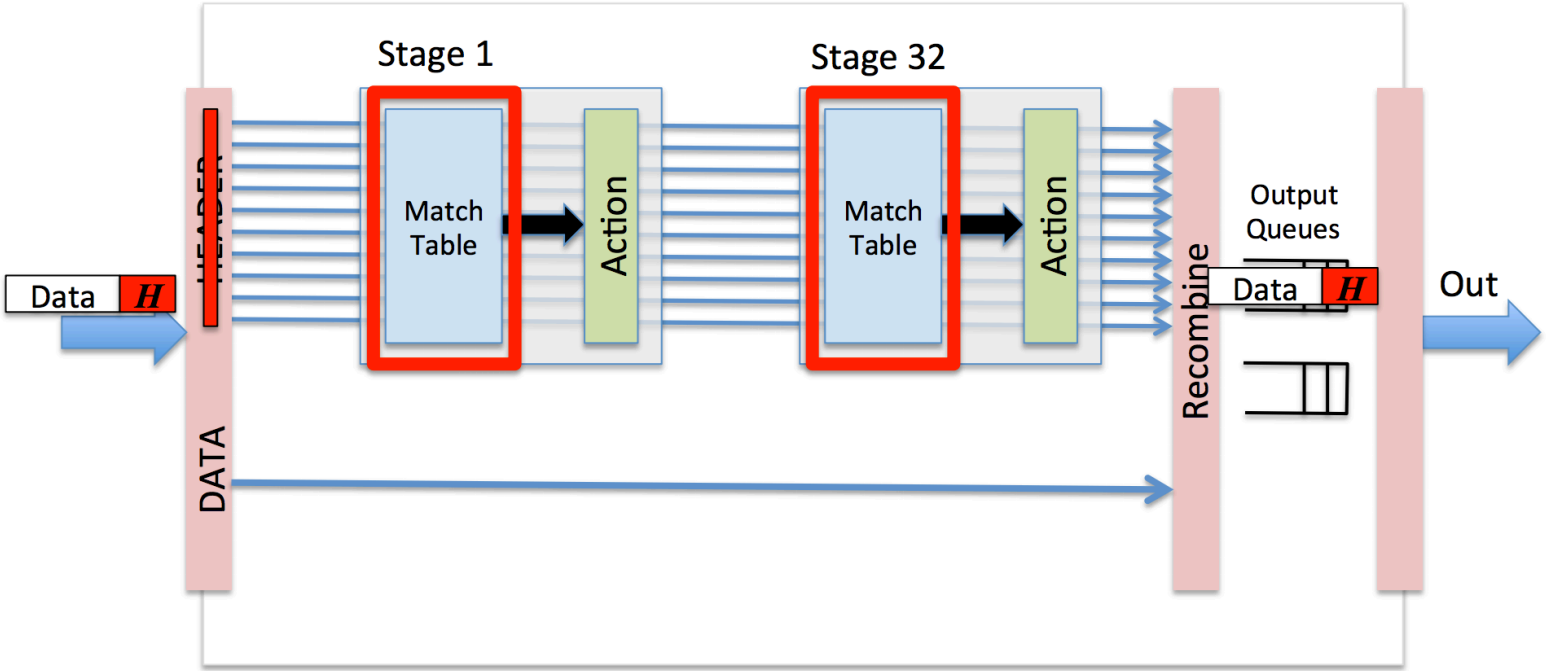
- An "OpenFlow Chip"
  - Generalizable, programmable match-action primitives
- SwitchBlade
  - Programmable, modularizable FPGA-based data plane

OpenFlow Chip

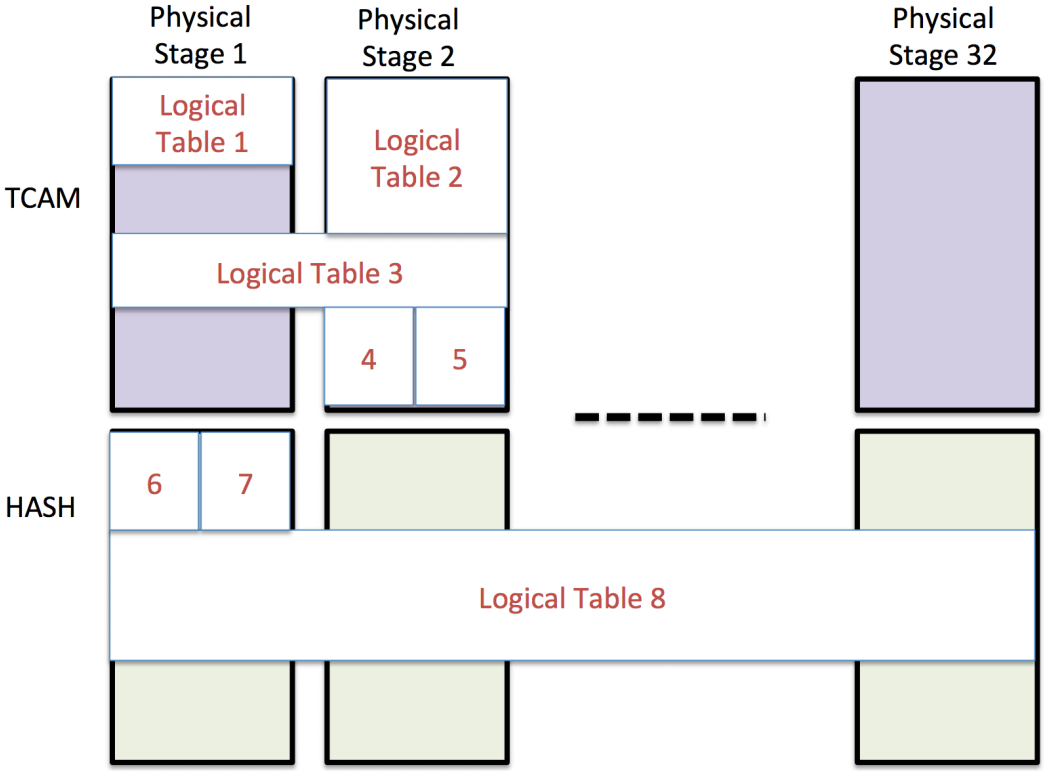
- 64 x10GE OpenFlow-optimized ASIC
- Industry-standard 28nm design process
- Parse existing + custom packet headers
- 32 stages of match-action
- Large tables (1M x 40b TCAM, 370 Mb SRAM)
- VLIW Action processing

Bosshart et al. "Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN", *ACM SIGCOMM*, August 2013.

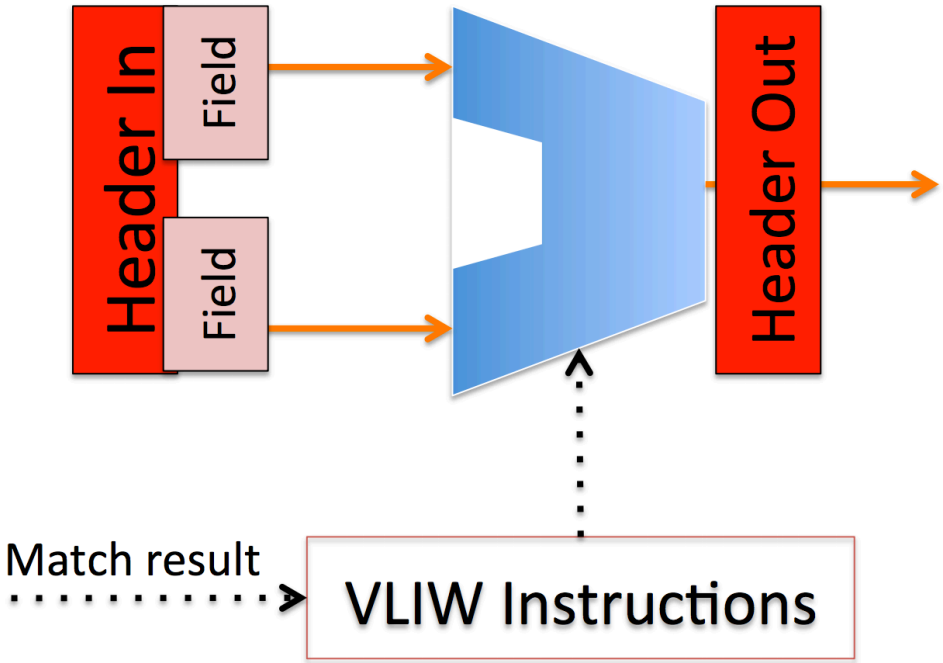
RISC-Like Architecture



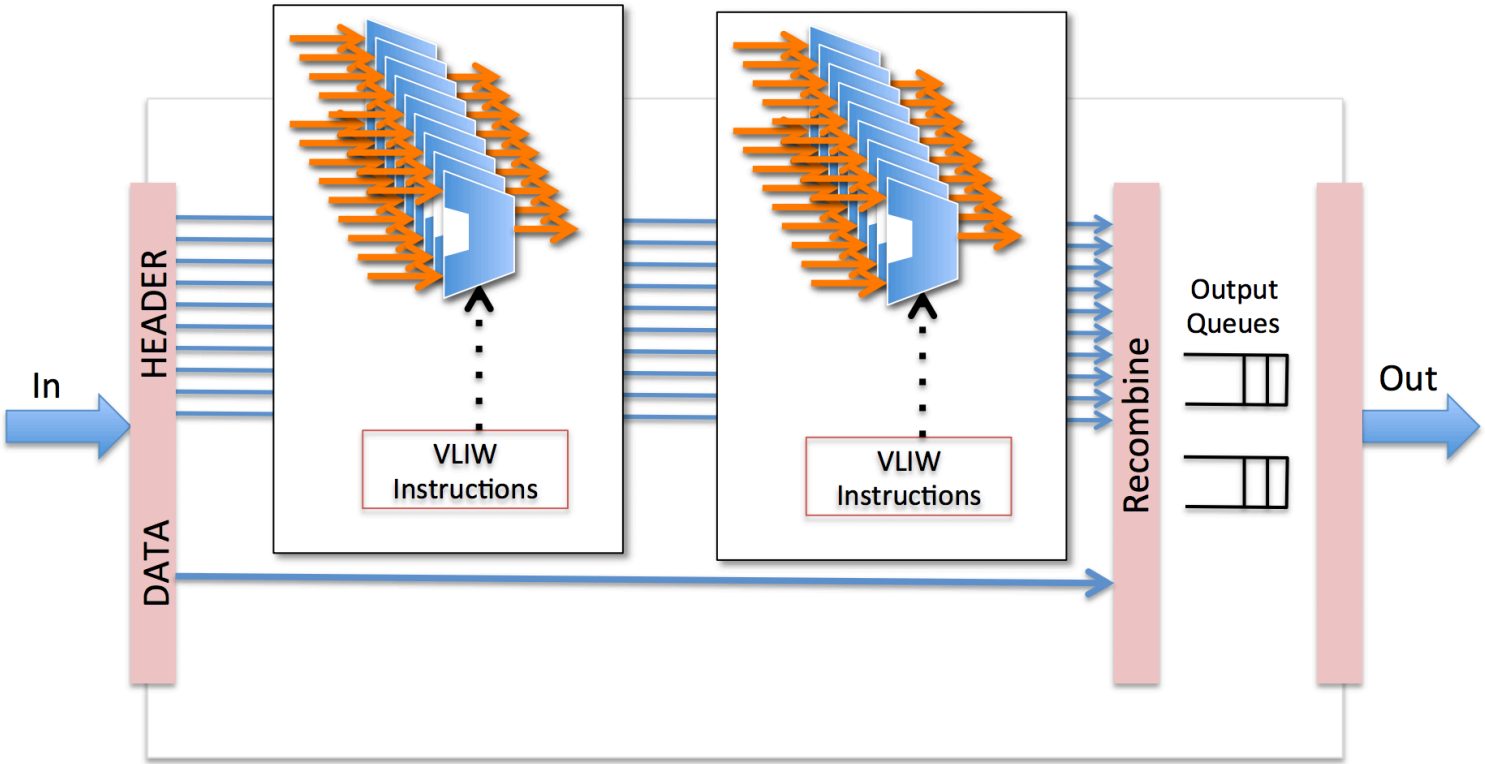
Flexible Match Tables



Action Processor



Actions Built into Stages



Two Examples of Modular Programmable Hardware Data Planes

- An "OpenFlow Chip"
  - Generalizable, programmable match-action primitives
- **SwitchBlade**
  - **Programmable, modularizable FPGA-based data plane**

SwitchBlade: Main Idea

- Identify **modular hardware building blocks** that implement a variety of data-plane functions
- Allow a developer to **enable and connect various building blocks in a hardware pipeline** from software
- Allow multiple custom data planes to operate **in parallel** on the same hardware

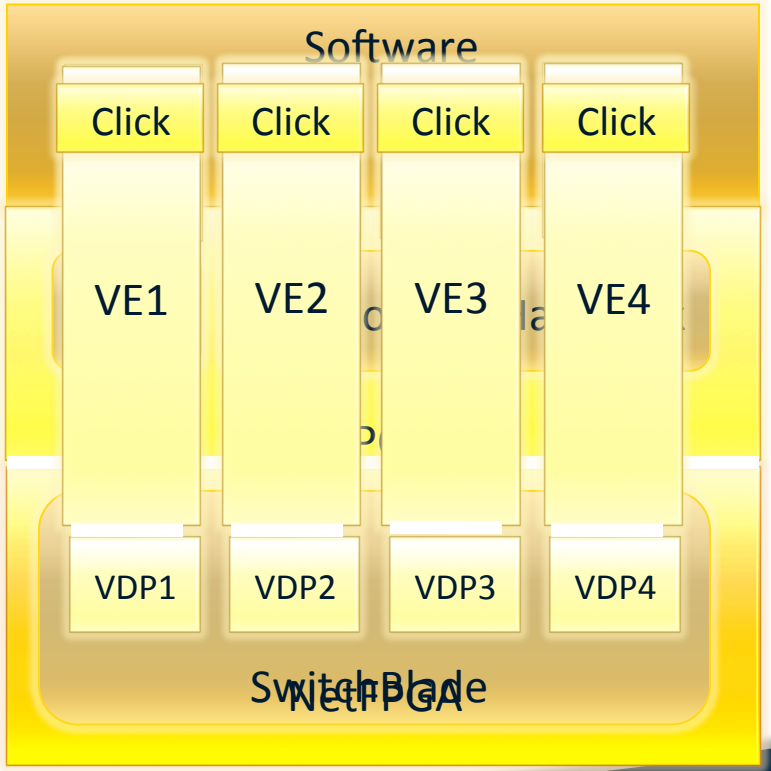
Flexible, fast, and easy to program.
Advantages of hardware and software with minimal overhead.

Anwer, Muhammad Bilal, et al. "Switchblade: a platform for rapid deployment of network protocols on programmable hardware." *ACM SIGCOMM Computer Communication Review* 40.4 (2010): 183-194.

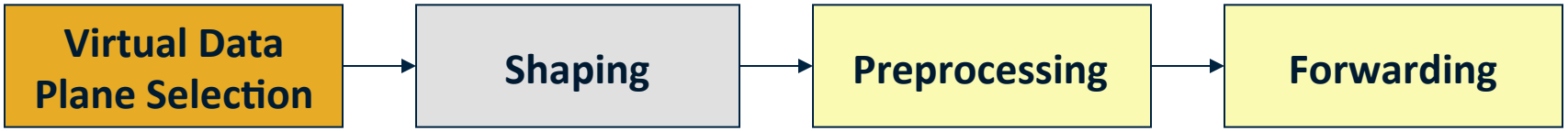
SwitchBlade: Push Custom Forwarding Planes into Hardware

Software
Hardware
Virtual Env.

VDP = Virtual Data Plane
Click = Click Software Router
VE = Virtual Environment



Virtual Data Planes (VDPs)

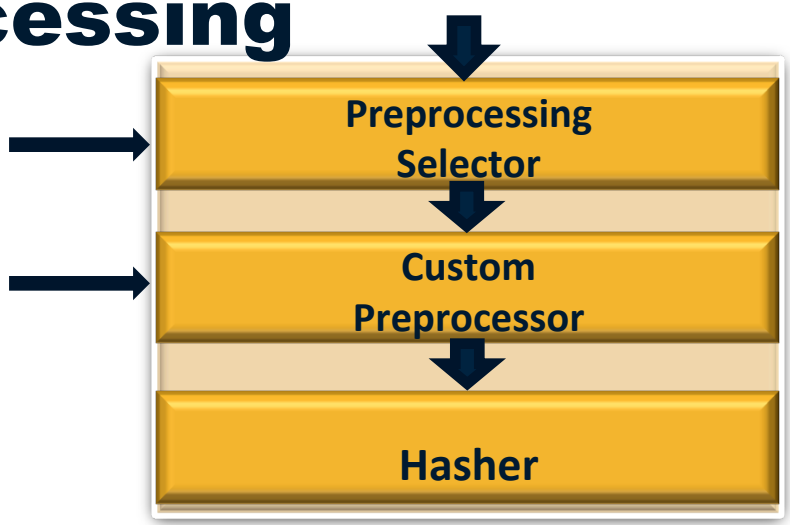


- Separate packet processing pipeline, lookup tables, and forwarding modules per VDP
- Stored table maps MAC address to VDP identifier
- VDP Selection step
  - Identifies VDP based on MAC address
  - Attaches 64-bit *platform header* that controls functions in later stages
  - Register interface controls this header per VDP

SwitchBlade Features

- Parallel custom data planes
  - Ability to **demultiplex** into existing data planes and maintain **isolation** on common hardware platform.
- Rapid development and deployment
  - **Pluggable preprocessor** modules to enable a range of customizable functions at hardware rates.
- Customizability and programmability
  - Dynamic **selection of modules**, and ability to operate in several different **forwarding modes**.

Preprocessing



- Select processing functions from **library of reusable modules**
  - Fast customization without resynthesis
  - Example implementations: Path Splicing, IPv6, OpenFlow
- **Hash custom bits in packet header** and insert value in *hash* field in platform header
  - Enables custom forwarding

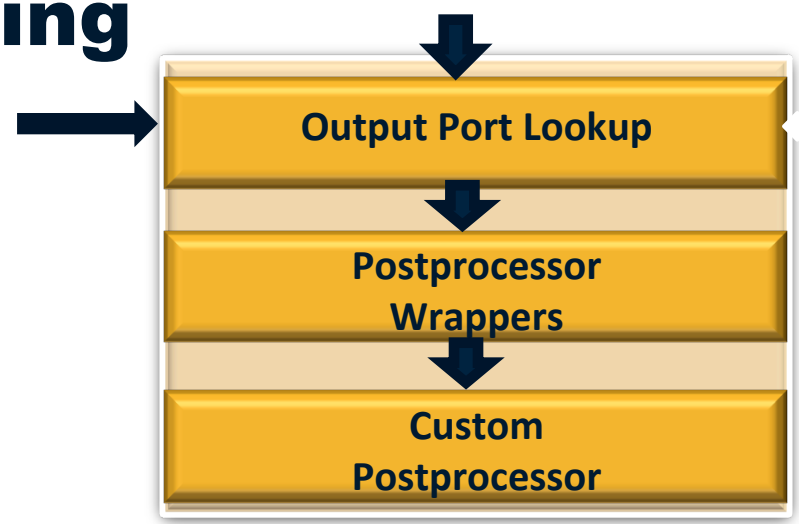
Example: OpenFlow

- Limited implementation (no VLANs, wildcards)
- Preprocessing Steps
  - Parse packet and extracts relevant tuples
  - 240-bit OpenFlow “bitstream” passed to **hasher** module in the preprocessor
  - Hasher outputs 32-bit hash value on which custom forwarding could take place
  - **Mode field** set to perform *exact match*

Adding New Modules

- Adding a new module at any stage requires Verilog programming
- User writes preprocessing (and postprocessing) modules to extract the bits used for lookup
- Resynthesize hardware
- Enable module from register interface in software

Forwarding

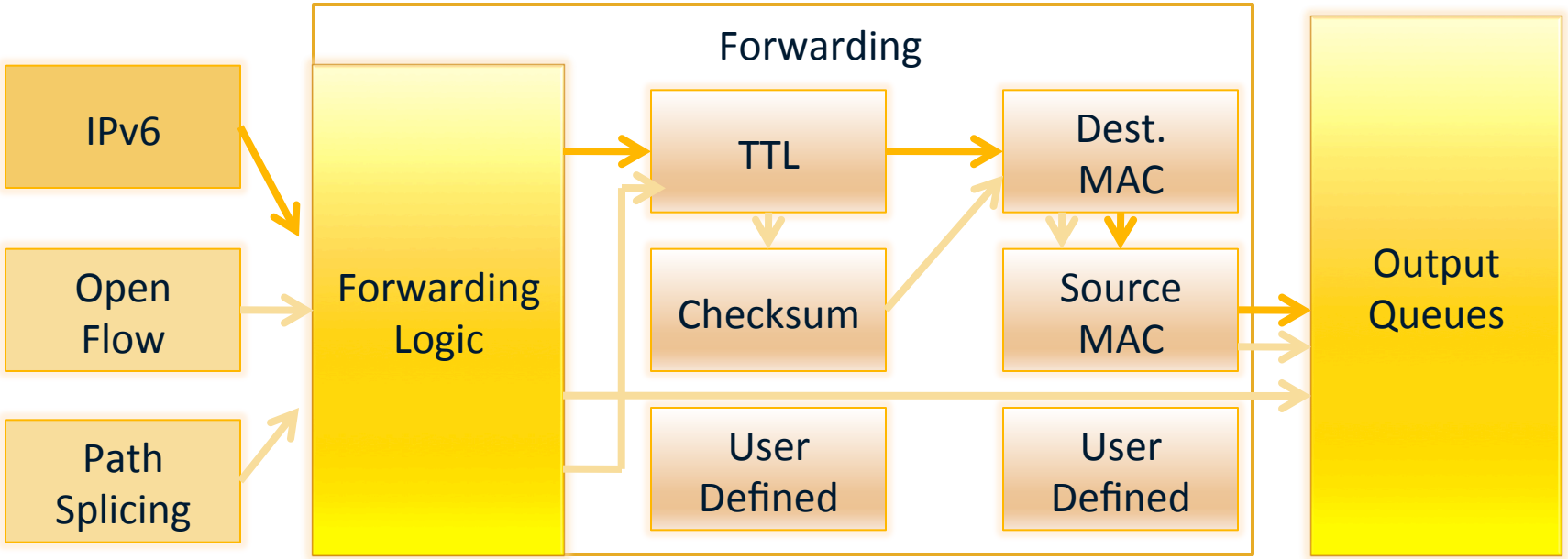


- **Output port lookup** performs custom forwarding depending on the mode bits in the platform header
- **Wrapper modules** allow matching on custom bit offsets
- **Custom post processors** allow other functions to be enabled/disabled on the fly

Software Exceptions

- Ability to redirect some packets to CPU
- Packets are passed with VDP (and platform header), to allow for VDP-based software exceptions

Custom Postprocessing Paths



Summary

- **Scalability: Make hardware programmable**
- Insight: Optimize a few primitives, provide composition
- OpenFlow Chip: Extreme flexibility, 15%+ area
- SwitchBlade: Programmable hardware, customizable data planes