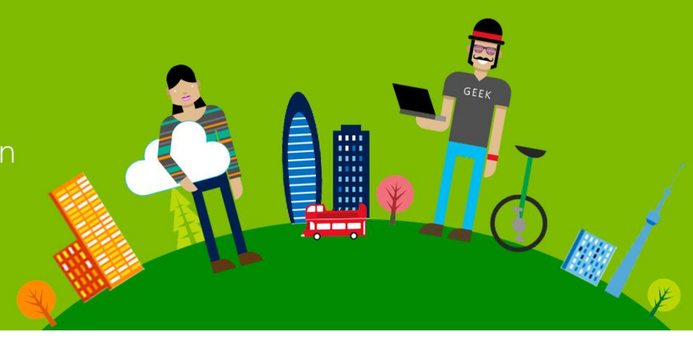
Microsoft Research Asia

Student TechFest 2016

Empower every person and every organization on the planet to achieve more.



ClickNP: Highly Flexible and High Performance Network Processing with Reconfigurable Hardware

Bojie Li, Kun Tan, Layong (Larry) Luo, Yanqing Peng, Renqian Luo, Ningyi Xu, Yongqiang Xiong, Peng Cheng
Wireless and Networking group

s16comm Florianópolis, Brazil
To appear in SIGCOMM'16

Specialization vs. Generalization

- **Specialized hardware**: efficient but not flexible
- **General-purpose processor**: slow but flexible
- **Reconfigurable hardware**: the sweet point between two extremes

Catapult FPGA from MSR NeXT (ISCA '14)

- FPGA is being deployed at scale in **MS data centers**
- **Accelerate cloud services** and save CPU: Bing search, Azure Networking, Azure Storage…

Challenge: FPGA programming

- **Low-level** hardware description languages: Verilog, VHDL…
- **Hard to program**, hard to debug

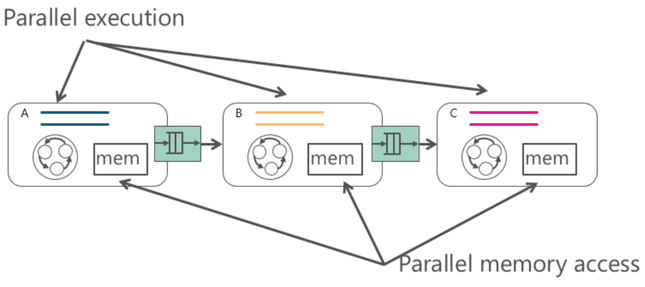
Fortunately, there are High-Level Synthesis tools

- Develop program in high-level programming language (**C-like**) and synthesize into hardware description languages
- However, may generate **surprisingly poor hardware** from C code

FPGA architecture is different from CPU

- **Slow clock frequency**: 200 MHz vs. 2~3 GHz (CPU)
- **Slow memory**: 2~4 GB/s vs. 40~100 GB/s (CPU/GPU)
- But: **Massive parallelism**
  - 172,600 logic elements
  - 2,014 memory blocks (each 20Kbit)
  - 1,590 DSP blocks
  - Each of them can work in parallel
- Efficient FPGA code must utilize this massive parallelism

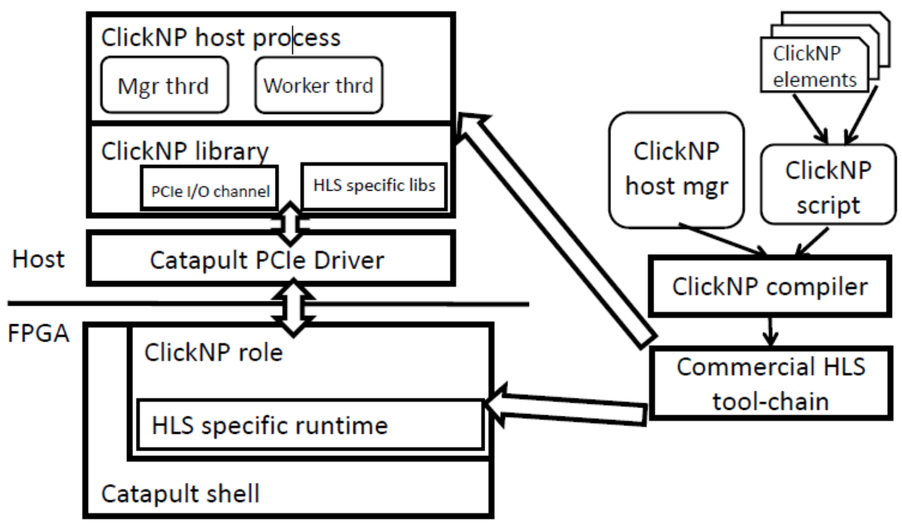
Streaming programming model



Design goals

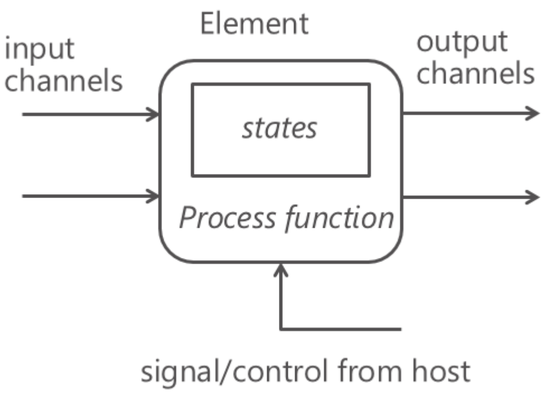
- **Flexibility**: fully programmed using high-level languages
- **Modularized**: modular architecture for packet processing
- **High performance and low latency**: 40 Gbps line rate, 2 microsecond latency for any packet size
- **Support joint CPU/FPGA packet processing**: Efficient split of work between CPU and FPGA

ClickNP system architecture



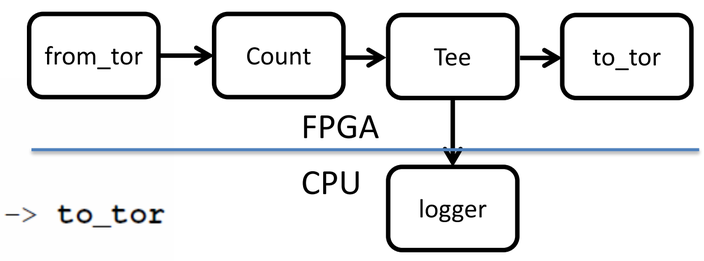
ClickNP element

```
1  .element Count <1, 1> {
2    .state{
3      ulong count;
4    }
5    .init{
6      count = 0;
7    }
8    .handler{
9      if (get_input_port() != PORT_1) {
10       return (PORT_1);
11     }
12     flit x;
13     x = read_input_port(PORT_1);
14     if (x.fd.sop) count = count + 1;
15     set_output_port(PORT_1, x);
16
17     return (PORT_1);
18   }
19   .signal{
20     ClSignal p;
21     p.Sig.LParam[0] = count;
22     set_signal(p);
23   }
24 }
```



ClickNP element graph

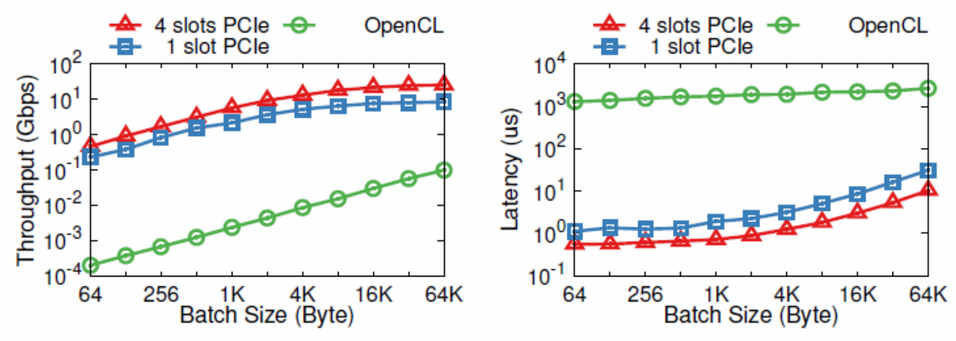
```
1 Count :: cnt @
2 Tee :: tee
3 host PktLogger :: logger
4
5 from_tor -> cnt -> tee [1] -> to_tor
6 tee [2] -> logger
```



Exploit parallelism inside element

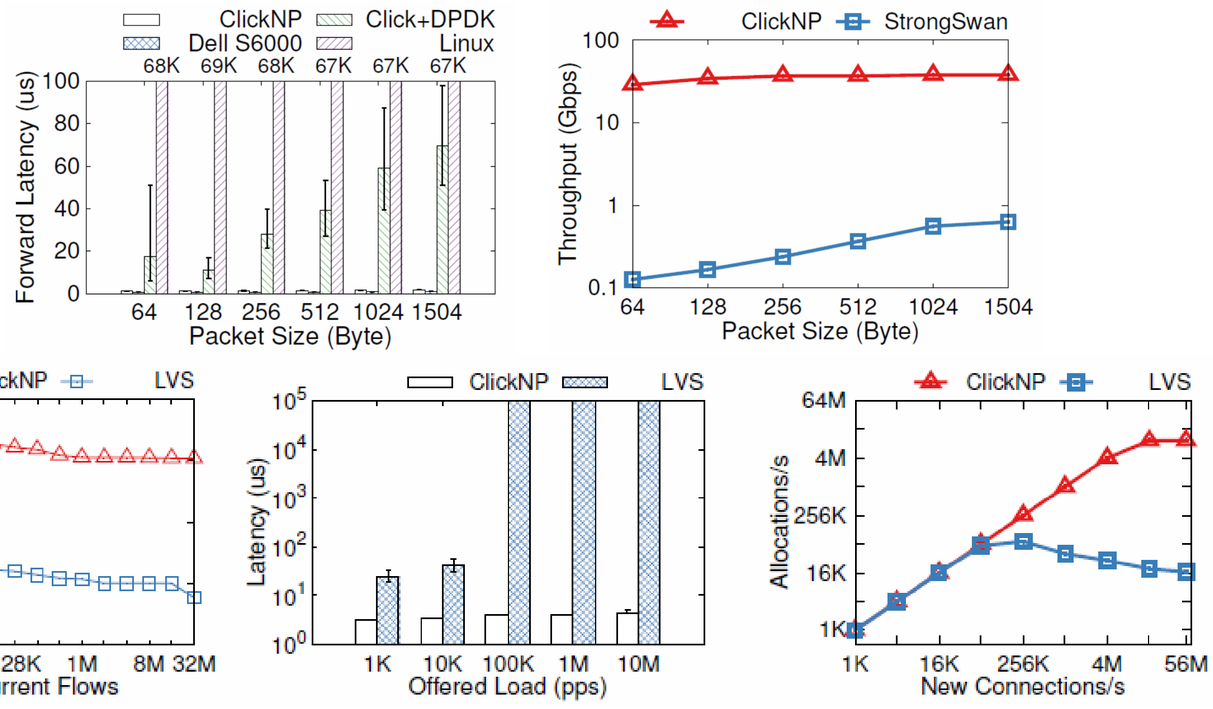
- Minimize memory dependency
  - **Use registers** for fast access
  - **Delayed write** to resolve read-write dependency
  - **Memory scattering** to remove pseudo dependency
- Balance pipeline stages
  - **Unroll loops** whenever possible
  - **Move slow branch** to a separate element

High-throughput and low-latency PCIe channel



5 applications built with ~100 elements

- **Packet generator and capture**: line rate at any packet size
- **Openflow firewall**: 1.23 us latency (similar to ASIC switch, 50x lower than CPU), 56.4 Mpps line rate (3x CPU)
- **IPSec gateway**: 46~200x throughput, 400x lower latency than CPU
- **L4 load balancer**: 32M concurrent flows, 10M new flows/sec
- **pFabric flow scheduler**: 4G strict flow priorities



Microsoft Research

ClickNP is the first FPGA-accelerated platform for network functions, written completely in high-level language and achieving 40 Gbps line rate at any packet size.