
Hunan Xperis Network

Queen Mary
University of London

FAST: Enabling Fast Software/Hardware Prototype for Network Experimentation

Xiangrui Yang, Zhigang Sun, Junnan Li, Jinli Yan, Tao Li, Wei Quan, Donglai Xu, Gianni Antichi

NUDT, Xperis Network, QMUL

yangxiangrui11@nudt.edu.cn

Outline

- **Motivation**
- **Programming Abstraction**
- **Implementation**
- **Evaluation**

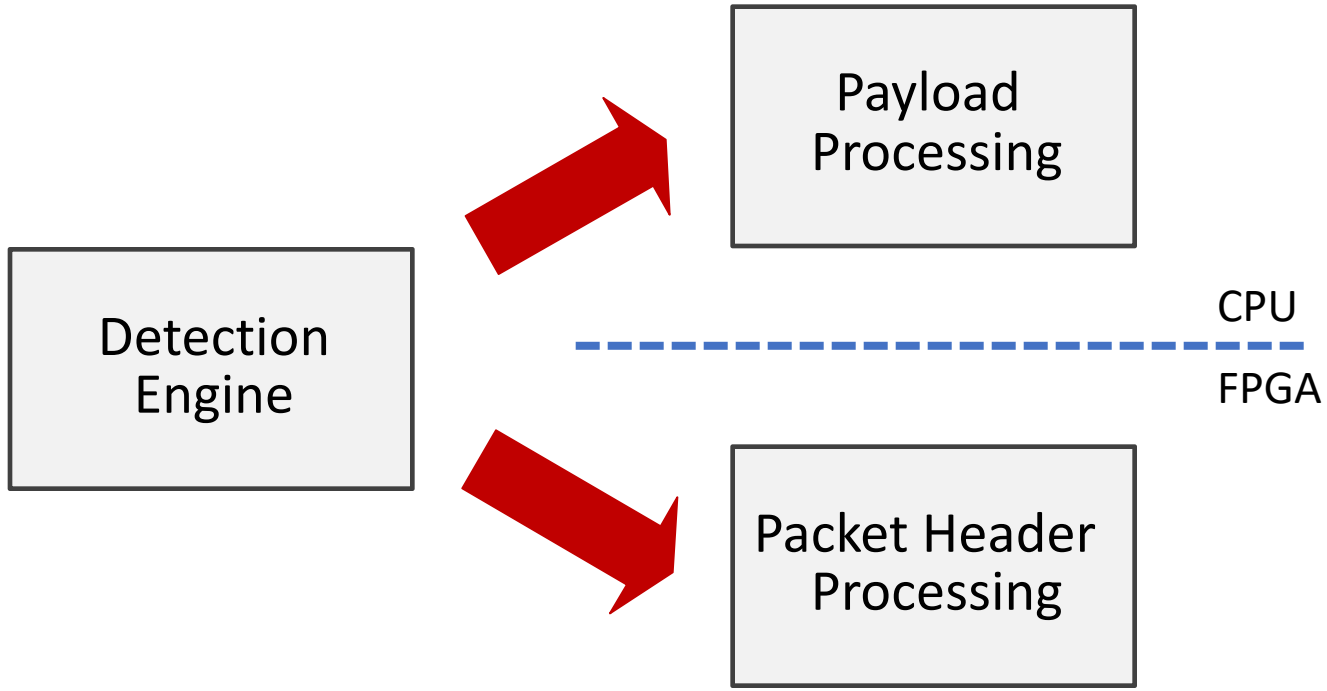
FAST is a novel software/hardware co-design framework for network experimentation

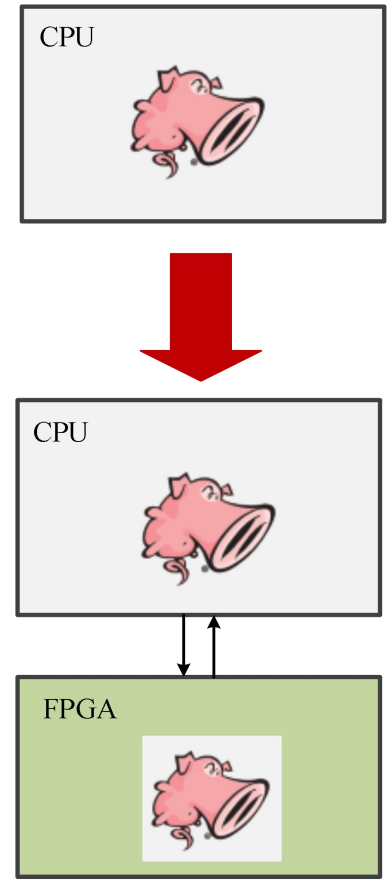
- Why is network experimentation with real systems important?
- What is wrong with the current network experimentation?

Why is CPU-FPGA co-design network experimentation important?

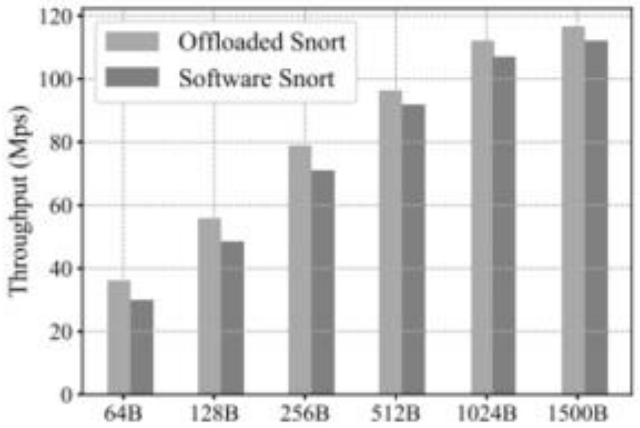
Take the **offloading of IDS** for example

- Offloading part of Snort functionalities onto hardware (FPGA) to gain **Performance** & save **CPU Resources**

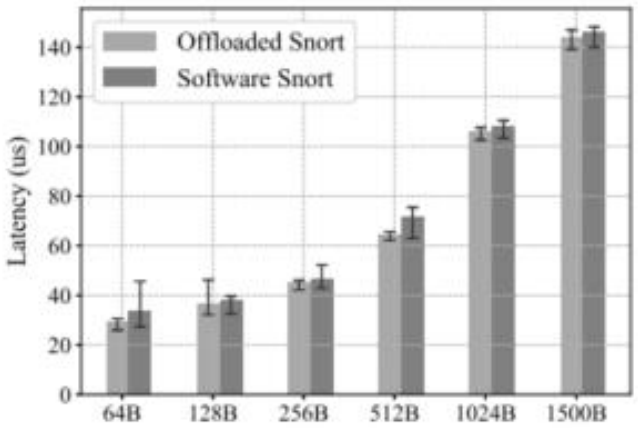




Offloading result..



(a) Throughput comparison



(b) Latency comparison

CPU: Cortex A9, 866MHz
FPGA: Artix-7, 125MHz
Offloaded Snort 2.02* vs baseline

	Idle Mode	Busy Mode
Offloaded version	10.402%	12.097%
Software version	10.386%	12.192%

Up to 8% in throughput and 3.5% in latency improvement observed. Very little performance gains regarding both throughput & Latency

A quick evaluation on real systems is very important

*Because Snort drops packets in high-speed, to measure the exact performance, we modify Snort to forward packets to a specified port after PacketCallback() function.

What’s wrong with the current network experimentation?

Build network prototype from scratch..

DMA

PCIe

AXI

Driver

MAC Block

Cross-Compile

Linux Kernel

System Calls

- Building a network prototype from scratch is more than complicated.
- It requires comprehensive knowledge from PCIe to NIC driver modification..

None of which is strictly related with network knowledge that researchers care

Using well-known platforms..

- Drawbacks:
  - **Bounded with specific platforms**
  - **Limited Supports on the SW**
    - NetFPGA (although it provides useful APIs on SW)
  - **Limited Supports on the HW**
    - DPDK, FIFO, VPP…

Core Problem

- There is no programming framework that targeting rapid network prototyping with CPU-FPGA platform of any kind

FAST aims to provide a unified framework for fast network experimentation with any CPU-FPGA platforms

Contributions

- Abstraction: A novel programming abstraction for fast network prototyping based on CPU-FPGA platforms
- Design: Comprehensive design details to support FAST abstraction on different platforms
- A number of opensource projects based on FAST to accelerate network prototyping

www.fastswitch.org

Design Options

- **Reduce development complexity**

  Reusable modules which are good for iterative development

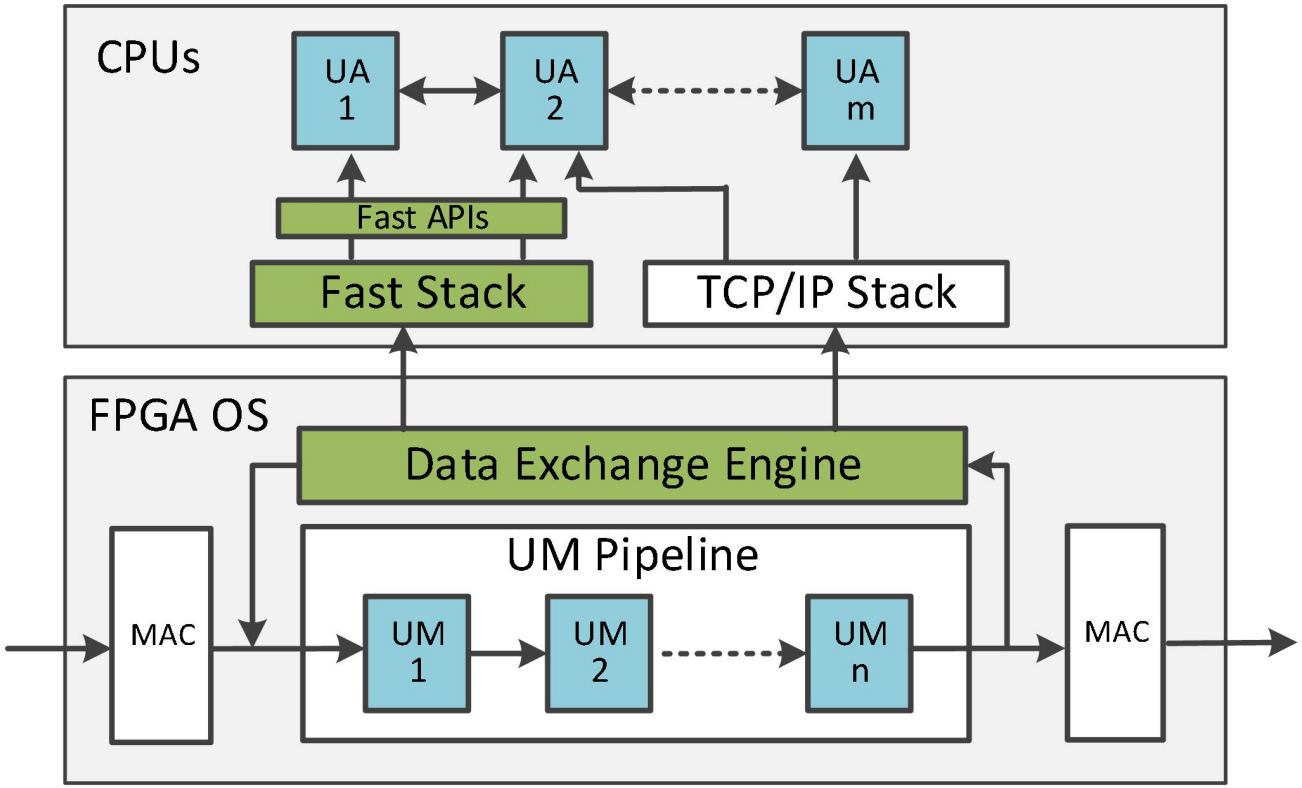
- **Clear & flexible path for packet processing**

  The same index mechanism for software module and hardware modules.

- **Unified APIs & data structures**

  easy-to-use APIs on the sw and metadata on the hw.

FAST framework



- Co-design framework based on CPU-FPGA co-processing architecture
- User-transparent Data Exchange Engine
- FAST libraries in software
- Scalable hardware pipeline

FAST Abstraction

Notion	Description
UA (User Application)	A piece of software code processing packet or control FPGA logic.
UM (User Module)	A hardware module that user writes to process packet.
MID (Module ID)	A specific ID to identify a UA or UM.
VAS (Virtual Address Space)	A range of virtual address for UA to access registers in UM.
MD (Metadata)	A piece of data carried before every fast packet to record info during processing.
FAST APIs	A set of APIs for UA to send/recv/ modify packets or read/write registers.

Each UA and UM has a specific ID

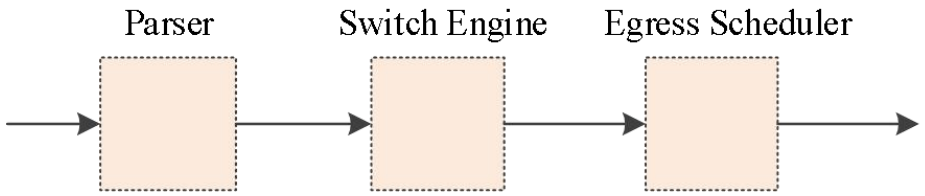
For UM, MID ranges from **0-127**;

For UA, MID ranges from **128-255**

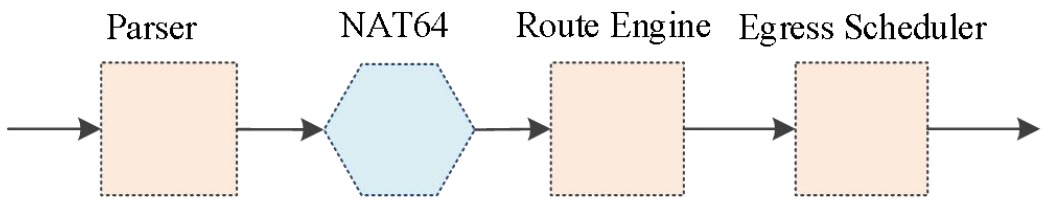
```
//ua register to the FAST stack
static int fast_ua_register(int mid);
//for sending packets to UMs
int fast_ua_send(struct fast_packet *pkt,int pkt_len);
//for read hardware registers from UA
u32 fast_ua_hw_rd(u8 dmid,u32 addr,u32 mask);
//for write hardware registers from UA
void fast_ua_hw_wr(u8 dmid,u32 addr,u32 value);
//provide similar functions like recv() in socket.
void fast_ua_recv();
```

Examples with FAST abstraction

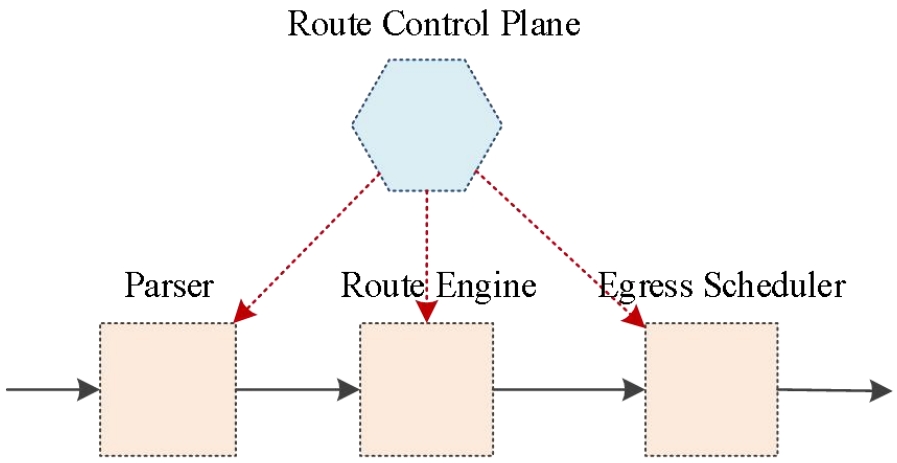
FAST supports 4 typical types of network prototype models and their combinations



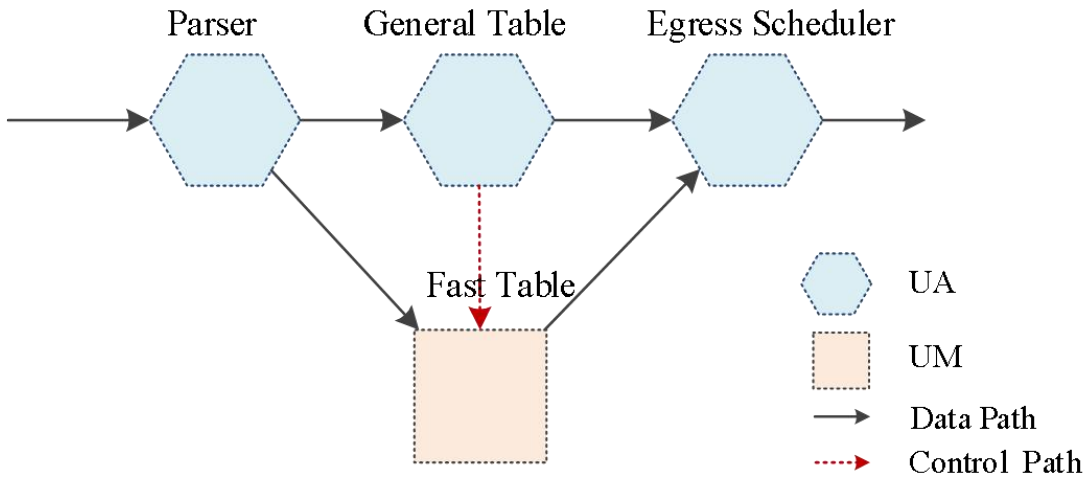
(a) Direct hardware processing.



(a) Software as assistance for packet processing.



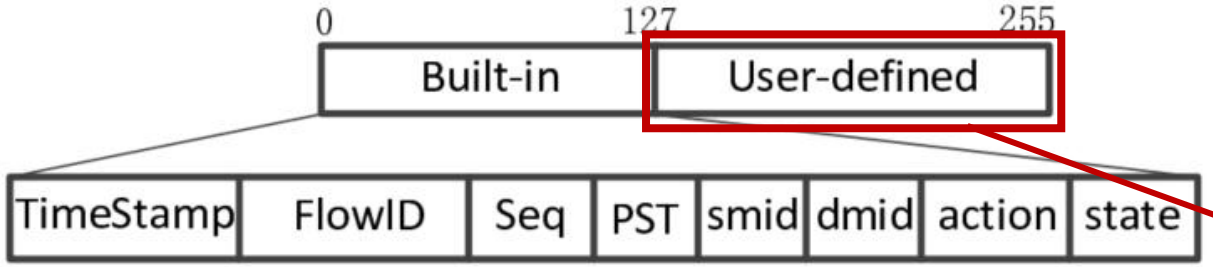
(c) Software as control plane for hardware processing.



(d) Hardware as fast path for software processing.

Key Data Structures

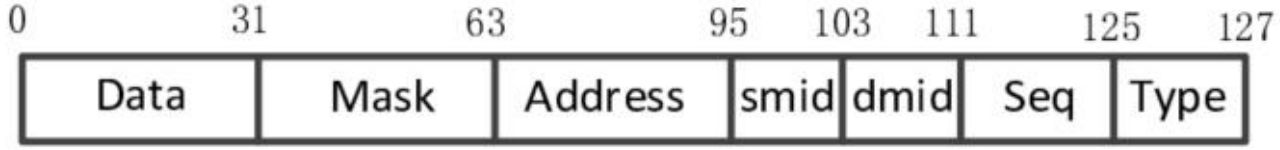
- Metadata



Metadata is attached to each packet while travelling across each module

This 128b field can to extended by users!

- FAST Control Packet

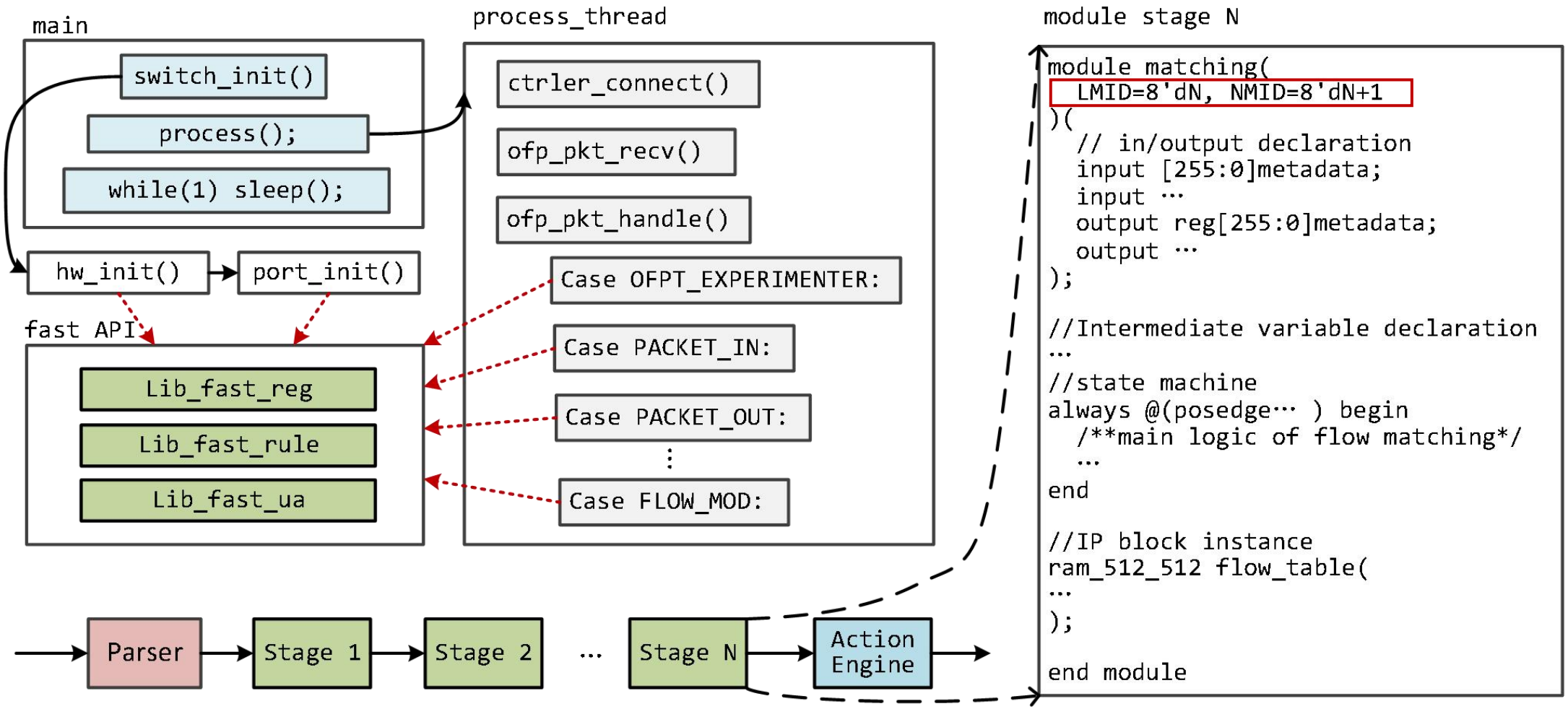


Type field

001: read
011: read response
010: write

FCP can be used to read/write registers of hardware by software

A FAST-based OpenFlow Switch..

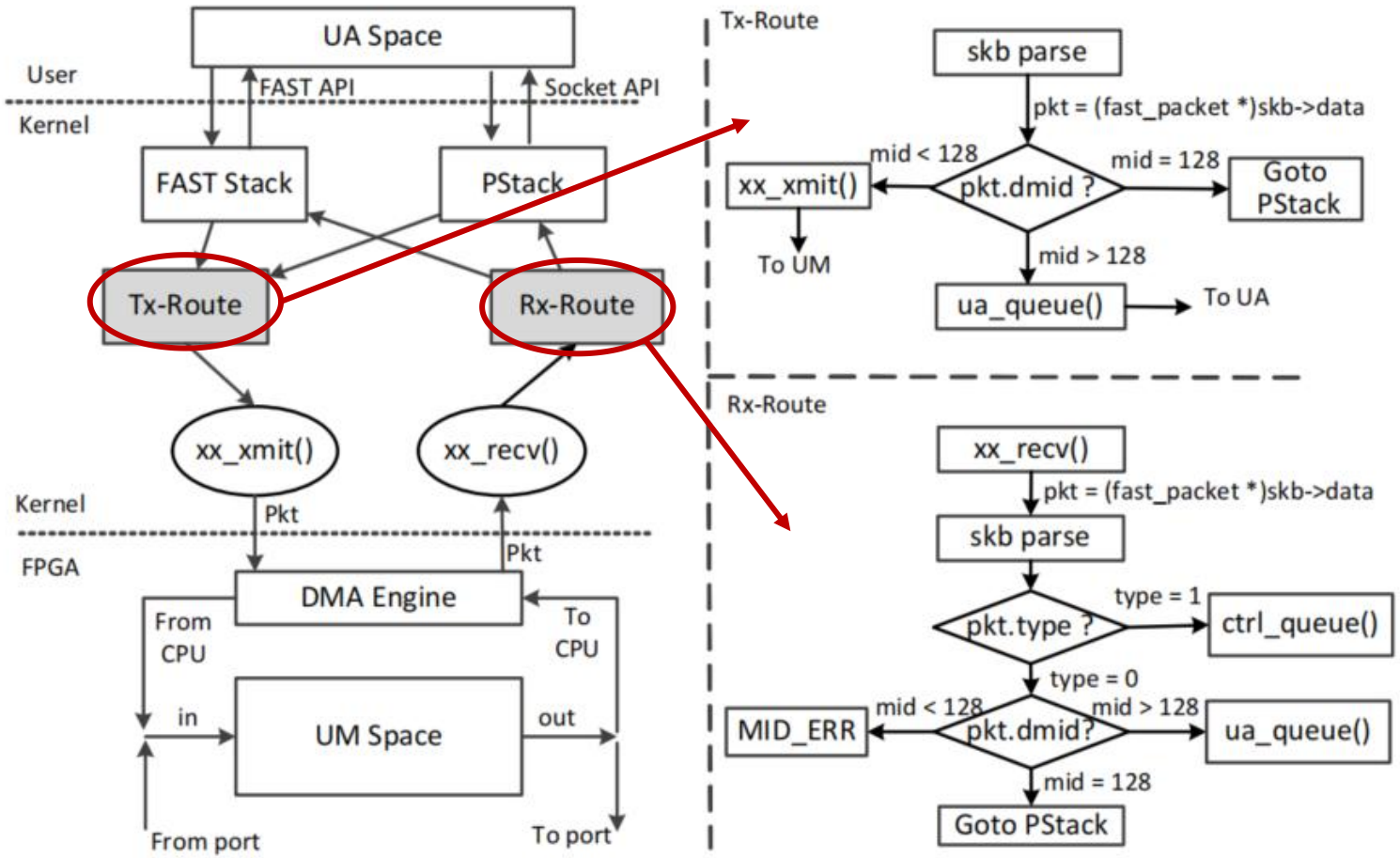


FAST Implementation

One of the key features of FAST is that it helps users to build network prototypes based on different types CPU-FPGA platforms

- Direct interconnection with PCIe (SoC..)
- Ethernet as interconnection (FPGA cloud..)

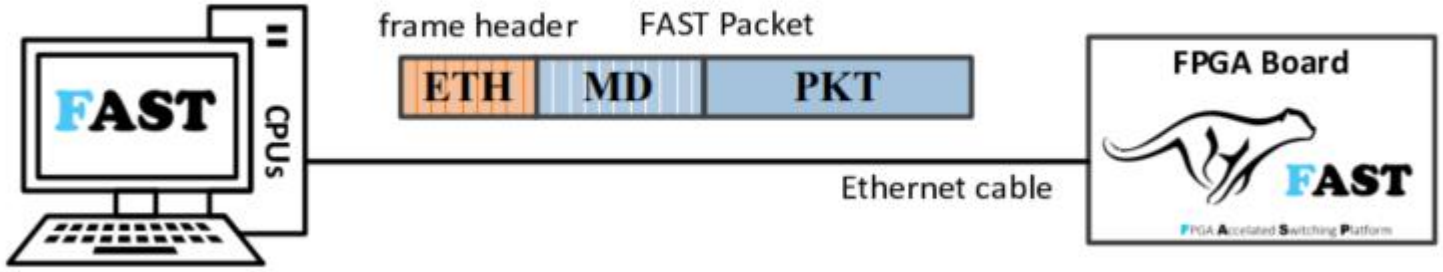
PCIe as interconnection



- Users Space: FAST Library
- Kernel Space: I/O drivers, TX/RX functions
- FPGA: DMA Engine (implemented by third party IP blocks)

Ethernet as Interconnection

By **encapsulation**, FAST can also be implemented on bare FPGA board with a remote PC/Server



E.g. MAC addr can be leveraged as the dest addr of the remote FPGA board. FAST packet can be encapsulated inside a L2 header

Evaluation

- Performance

  Will FAST introduce great throughput or latency cost in the prototype?

- Usability

  Can FAST support various network prototype design?

- Feasibility

  Can the prototypes based on FAST satisfy the performance requirements?

FAST 1GE Evaluation

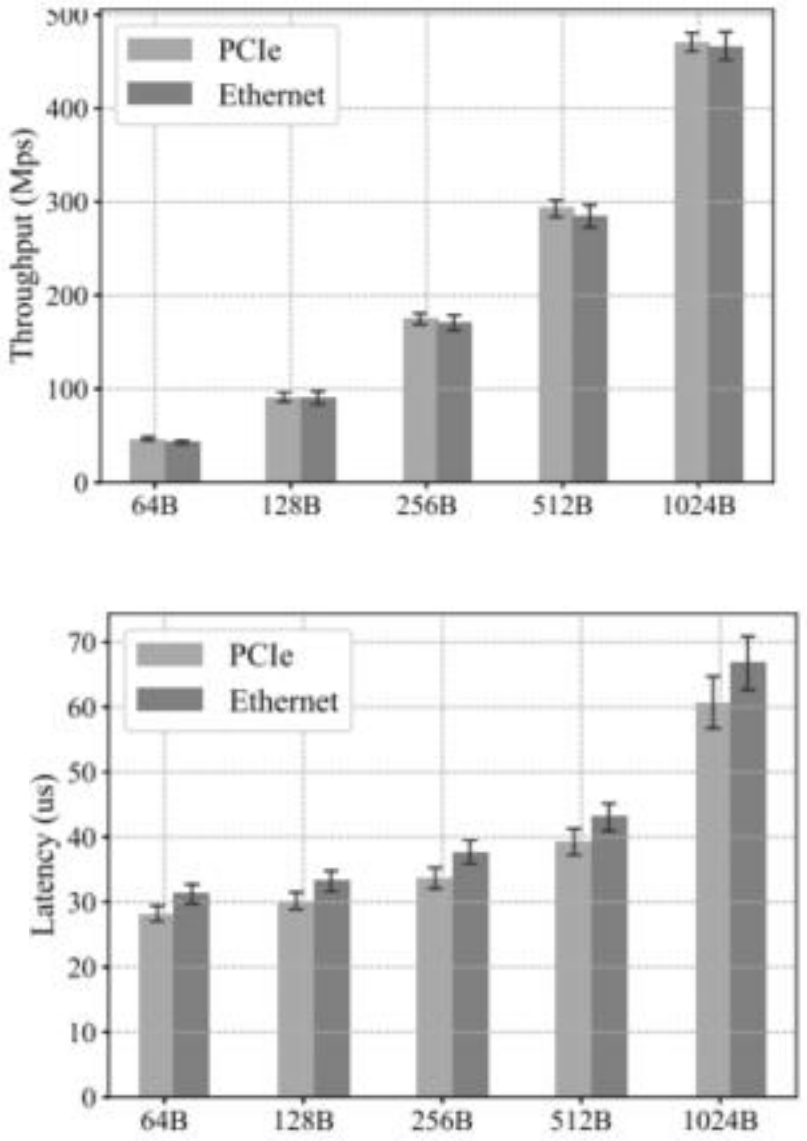
Compare the throughput & latency between CPU & FPGA between 2 types of CPU-FPGA platforms:

CPU: ARM Cortex-A9, 866MHz

FPGA: Aritix-7, 125MHz (with a default pipeline)

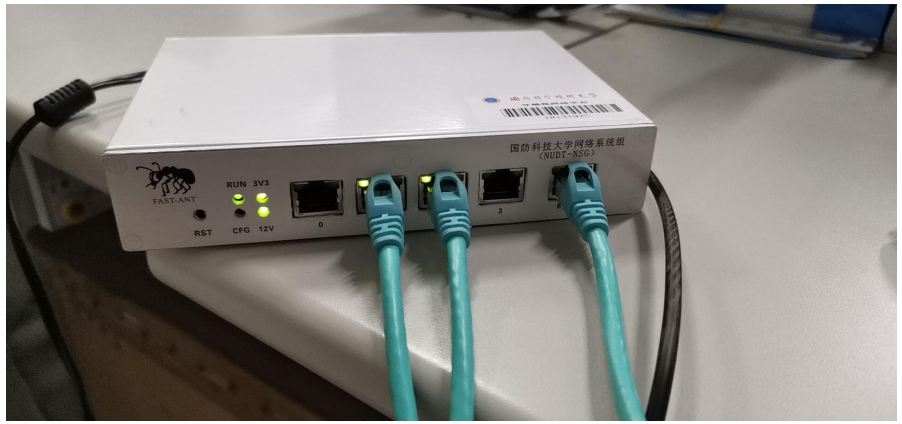
tools: using **IXIA PerfectStorm** for testing

Using Netlink for communication between UA and kernel space

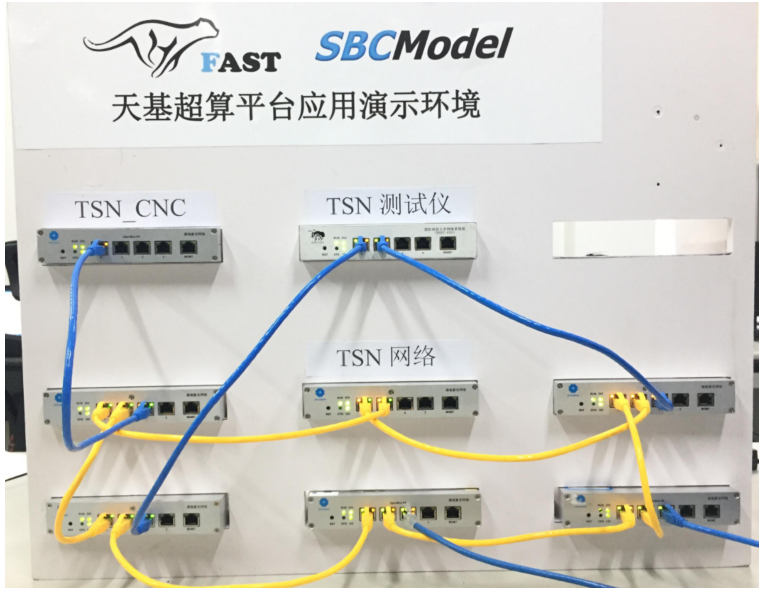


Projects based on FAST

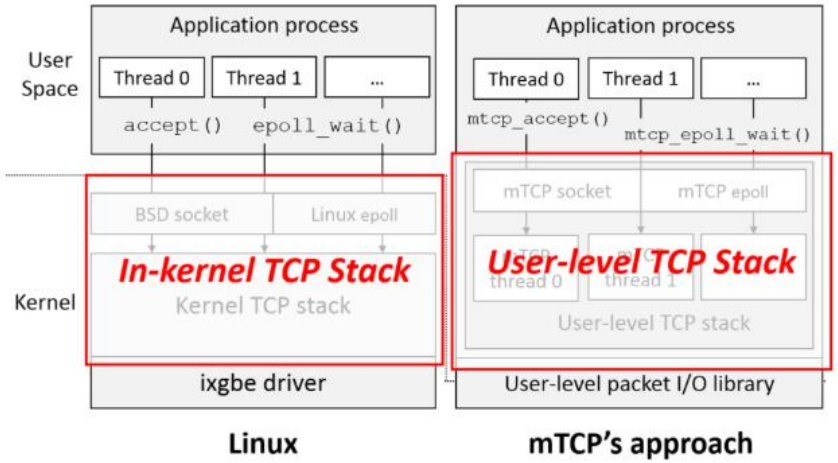
ANT (Agile Network Tester)



FAST-TSN Platform



UniMon (mOS + FPGA)



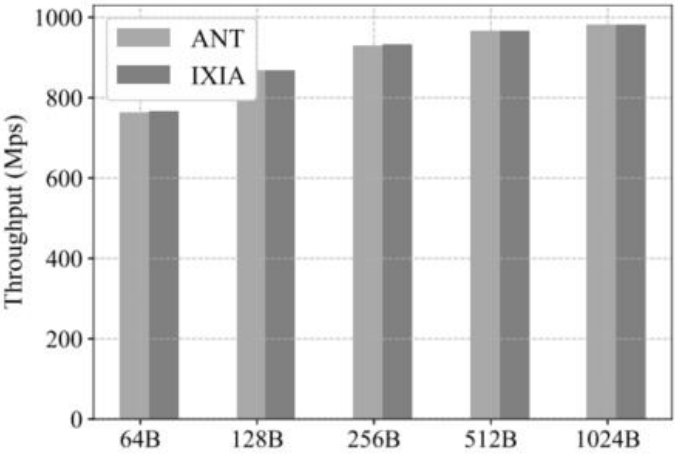
FAST-XTR Gateway

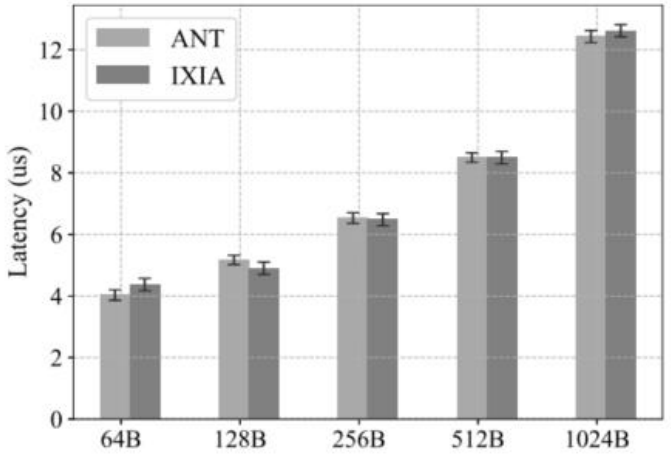
To be continued…

Some of the results..

Tester: **IXIA PerfectStorm tester** vs **ANT**

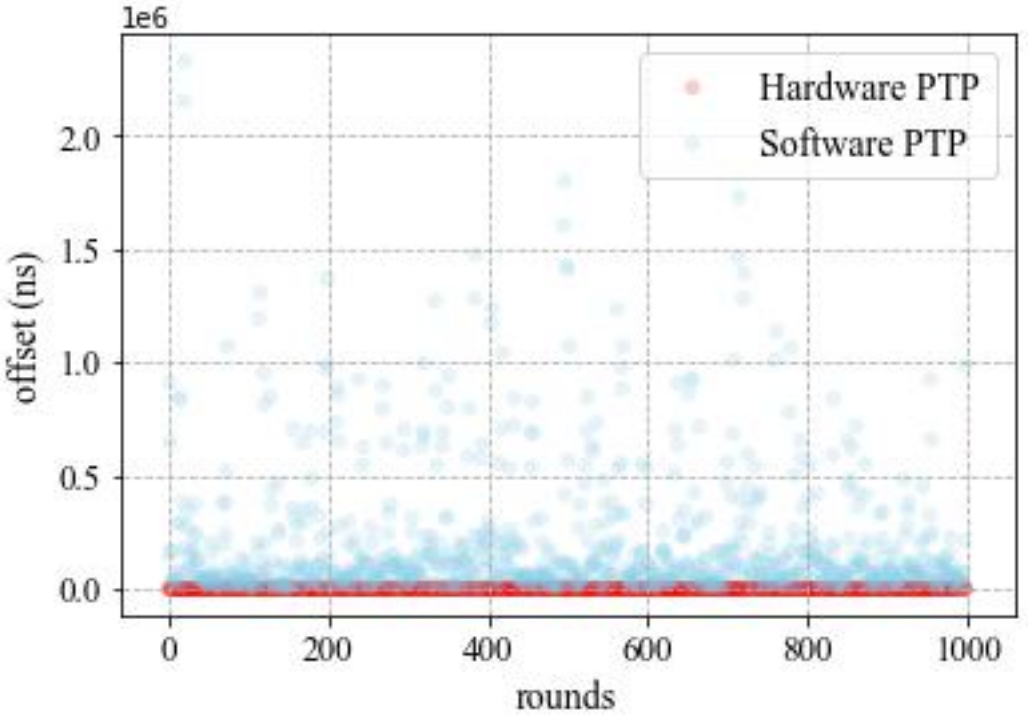
Tested: **Pica8 P3297**





HW PTP (FPGA) vs **SW PTP**

Time Synchronization: **PTP**



FAST provides almost the same accuracy for ANT compared with typical commercial testers.

Getting more..

Collaborators

NUDT & Xperis Network

Join FAST Community

Follow us and join FAST community on Wechat !

E-mail: nudtyxr@hotmail.com

Visit us on www.fastswitch.org

Q & A

- Please contact me: yangxiangrui11@nudt.edu.cn or nudtyxr@hotmail.com

Thanks !