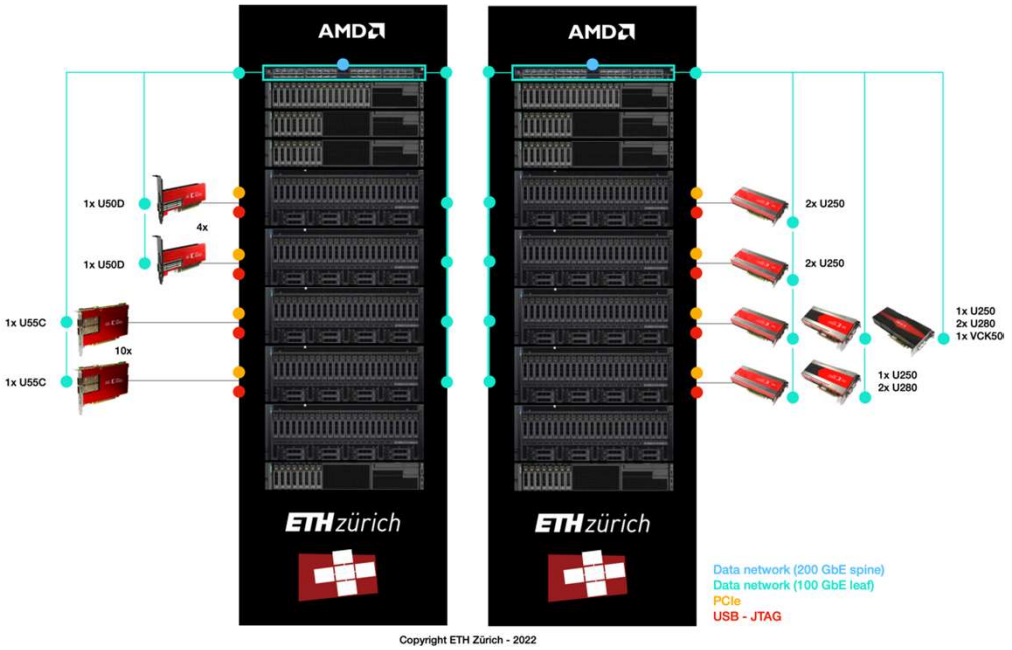
Systems@ETH Zürich

EasyNet: 100 Gbps TCP Support for Vitis

Zhenhao He



EasyNet Goal

Integrate a 100 Gbps TCP/IP stack into Vitis platform

- Easy to instantiate
- Low latency & high throughput

Provide HLS API for communication

- Take advantage of HLS
- Point-to-point / collective communication

TCP Network Offload Roadmap

Systems@ETH zürich

Years of efforts on TCP stack offload

- 10 Gbps TCP, FCCM'15, FPL'16
- 100 Gbps TCP – Limago, FPL'19
- **100 Gbps TCP for HLS – EasyNet**, FPL'21

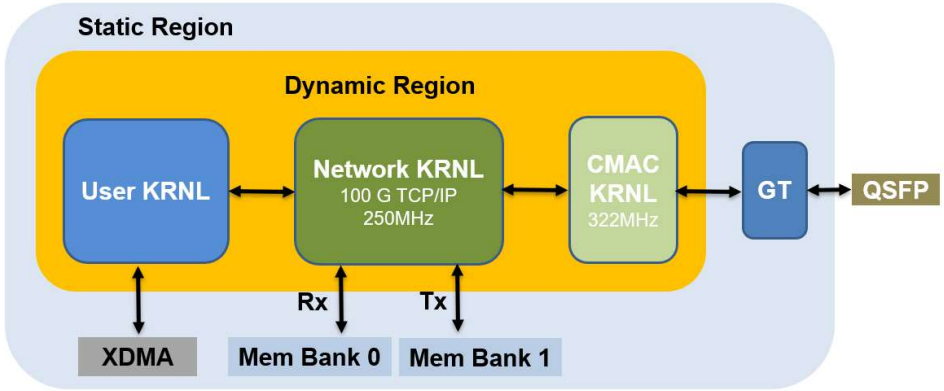
Improvement over years and feature complete

- Line-rate processing
- Retransmission
- Window scaling
- Out-of-order
- Thousands of connections

Tested and used in many projects

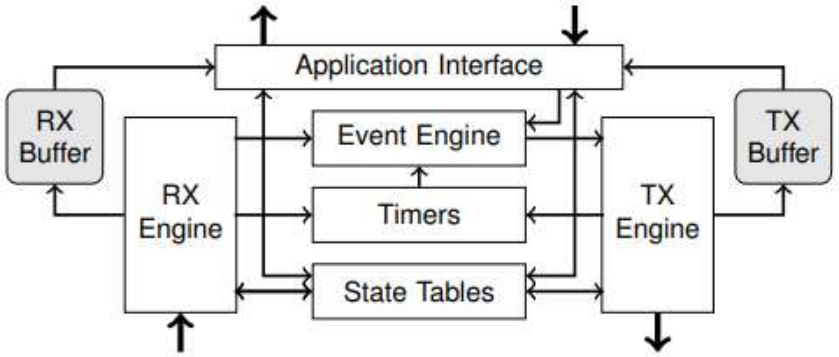
- Smart scatter-gather, SoCC'20
- Distributed Recommendation, KDD'21, FPL'21

Overall Architecture



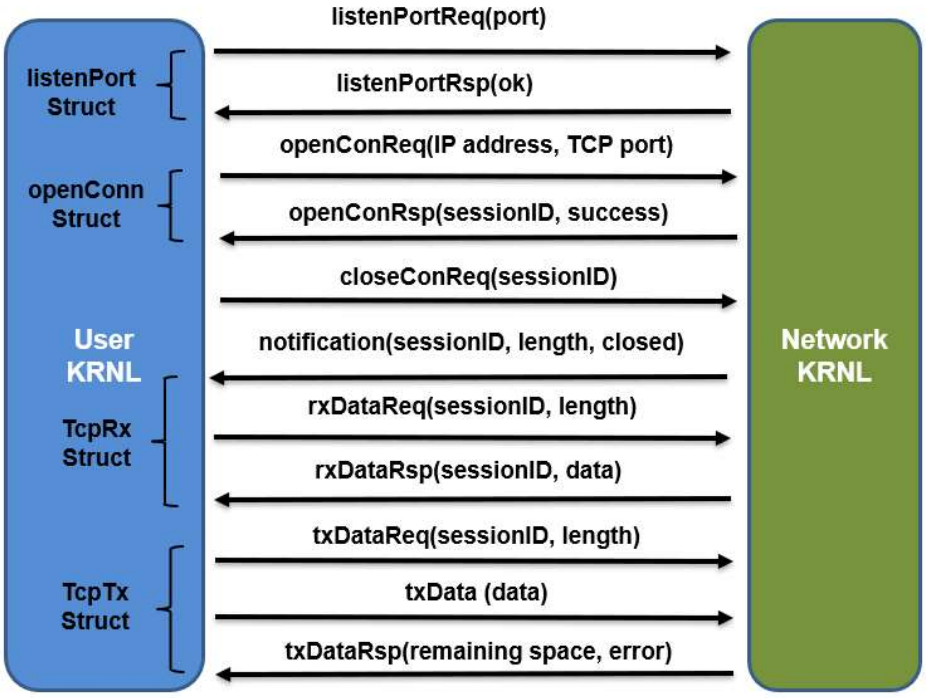
- CMAC Kernel
  - Ethernet subsystem, board specific
- Network Kernel
  - TCP/IP stack with streaming interfaces
- User: Customized unit for application
  - HLS and RTL kernel support

Network Kernel Architecture



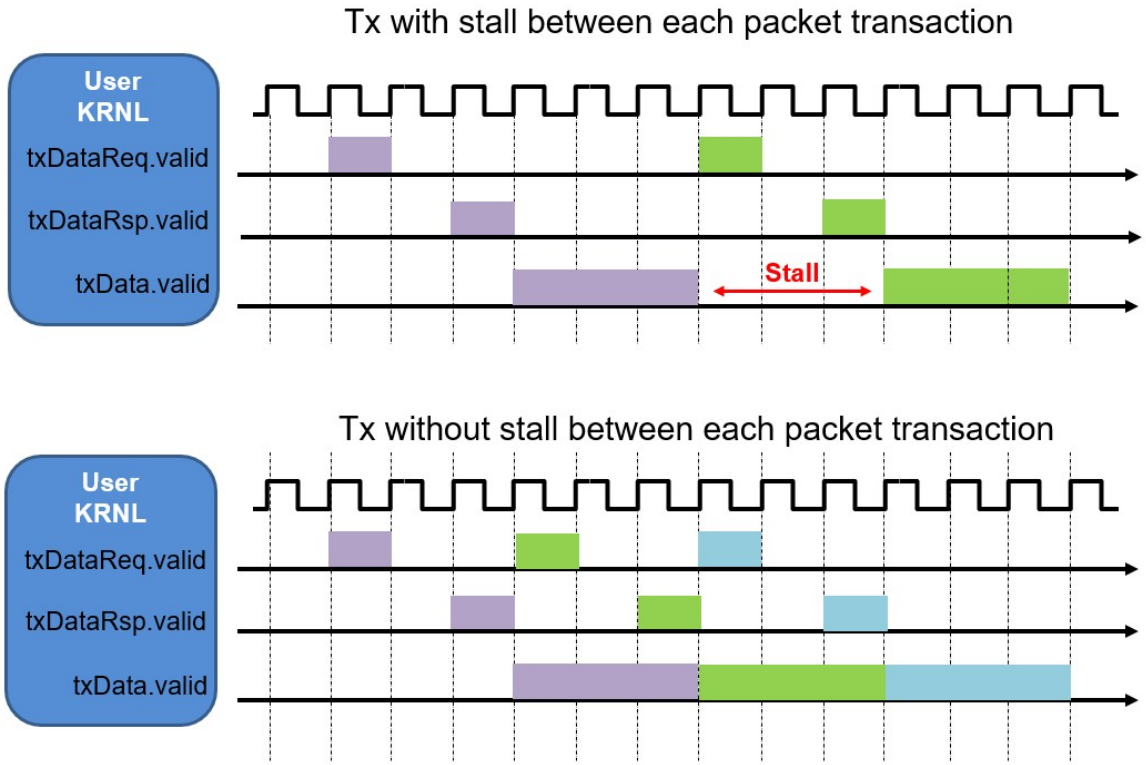
- Streaming interfaces to User and CMAC kernel
- Session states kept in BRAM
- Tx buffer for retransmission
  - 64 KB buffer per session
  - In DDR/HBM
- Rx buffer for re-ordering
- ICMP and ARP

User – Network Kernel Streaming Interfaces



- Handshake protocol
  - Port and connection handshake
  - Tx handshake
  - Rx handshake

Pipelined Handshake to Achieve High Performance



HLS Primitives to Abstract Low-level Handshakes

- HLS send

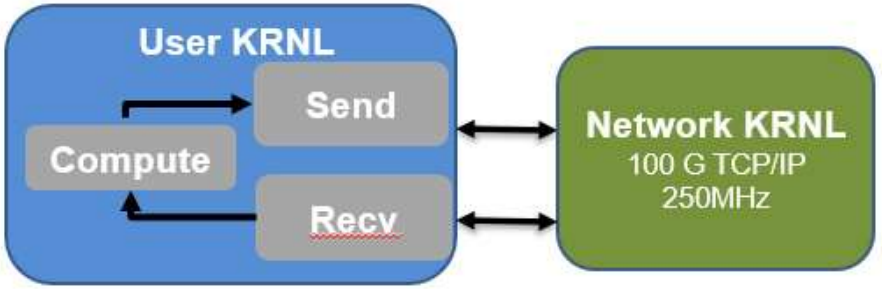
```
void send(dataType* send_data, uint64_t txByte,
    SessionStruct session, TcpTxStruct& TcpTxIntf);
```

- HLS receive

```
void recv(dataType* recv_data, uint64_t rxByte,
    SessionStruct session, TcpRxStruct& TcpRxIntf);
```

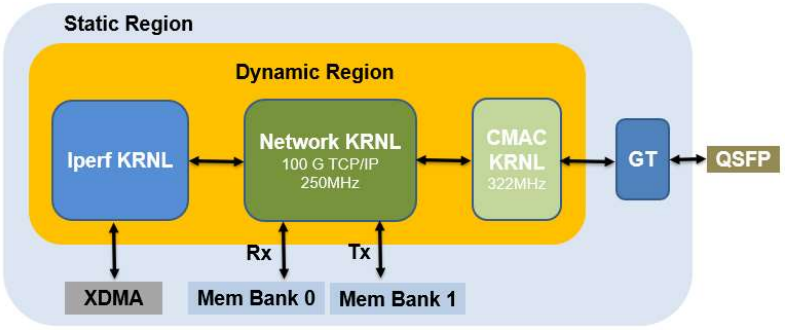
- Streaming & data flow

HLS Compile
-> Hardware



Example Kernels & Configurations

- User kernel supports both HLS and RTL kernels
  - HLS send, HLS recv
  - RTL iperf
- TCP configuration
  - TCP_STACK_MAX_SESSIONS
  - TCP_STACK_RX_DDR_BYPASS_EN
  - TCP_STACK_WINDOW_SCALING_EN
- Host bindings
  - OpenCL binding
  - Xrt Native
- Supported boards:
  - U280, U250, U55C

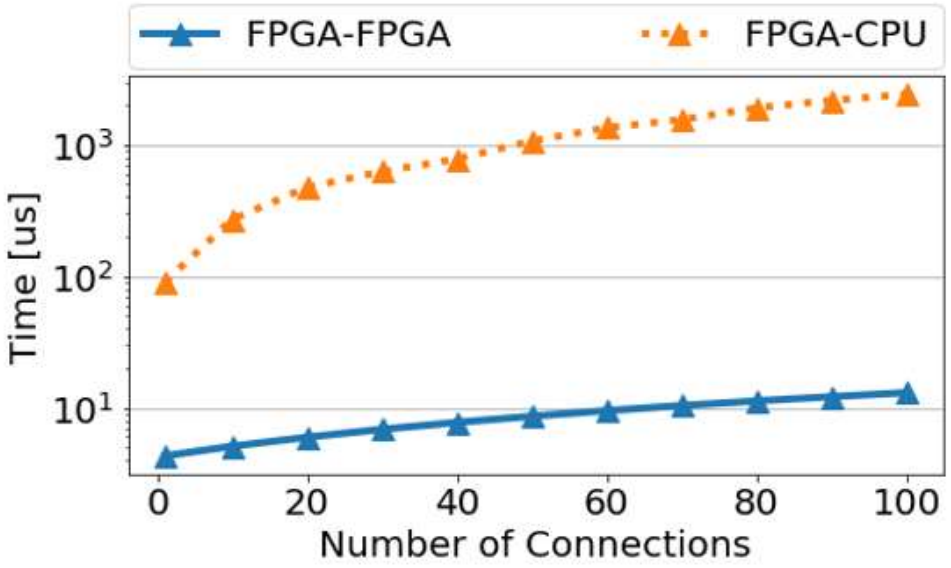



```
mkdir build
cd build
cmake .. -DFDEV_NAME=u280 -DTCP_STACK_EN=1 -DTCP_STACK_RX_DDR_BYPASS_EN=1
make installip
```

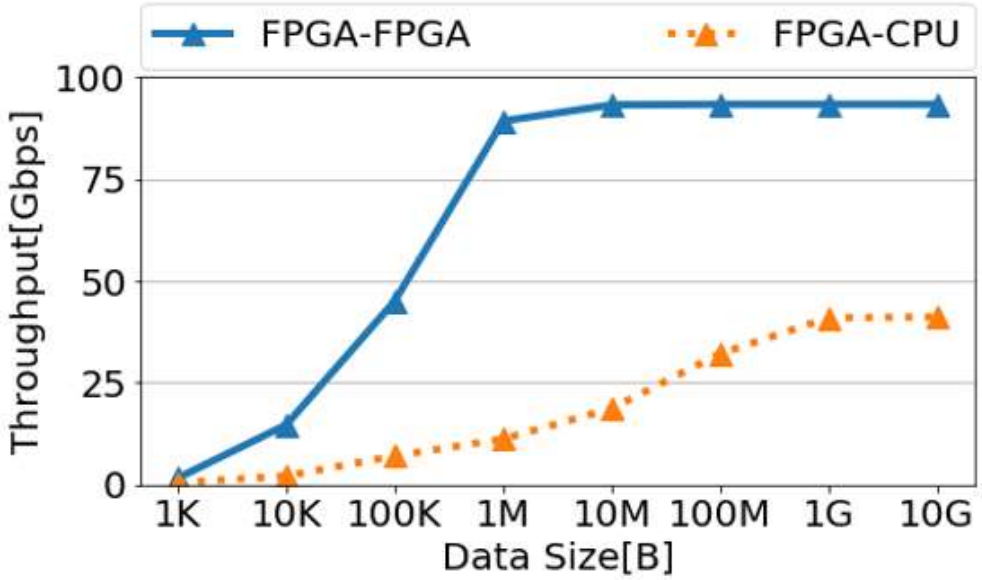
Performance

U280 FPGAs, interconnected via 100 Gbps Cisco Nexus network switch
Intel Xeon Gold 6234 processors with 100 Gbps Mellanox NIC
Performance measured in hardware

Latency RTT : FPGA-FPGA 5 us VS FPGA-CPU 46 us



Open Connection Time



Send & Recv Primitive Throughput

Resource Consumption

- Resource percentage on alveo-u280
- Most resource consumption by the network stack
- Ample room left for user application

TABLE II: Resource consumption

Resources	LUT	BRAM	DSPs
CMAC	15,717 (1.21%)	18 (2.24%)	0 (0%)
Network	114,699 (8.80%)	417 (7.09%)	0 (0%)
Send	2,326 (0.22%)	4 (0.29%)	0 (0%)

More information can be found:
https://github.com/fpgasystems/Vitis_with_100Gbps_TCP-IP