UNIVERSITY OF CAMBRIDGE

# P51: High Performance Networking

## Lecture 4: High Throughput Devices

Noa Zilberman
noa.zilberman@cl.cam.ac.uk

Lent 2019/20

# Project – Next Milestone

Due: Tuesday, 17/02/19 16:00

- Updated architecture
- Performance profile, including (but not limited to):
  - 10G interconnect
  - 10G module
  - Data path (no logic)
  - Architecture specific - e.g., memory access, recuirculation, externs etc.
- Expected overall speedup and throughput
- Memory requirements (if applicable)

# Very High Throughput Switches

# The Truth About Switch Silicon Design

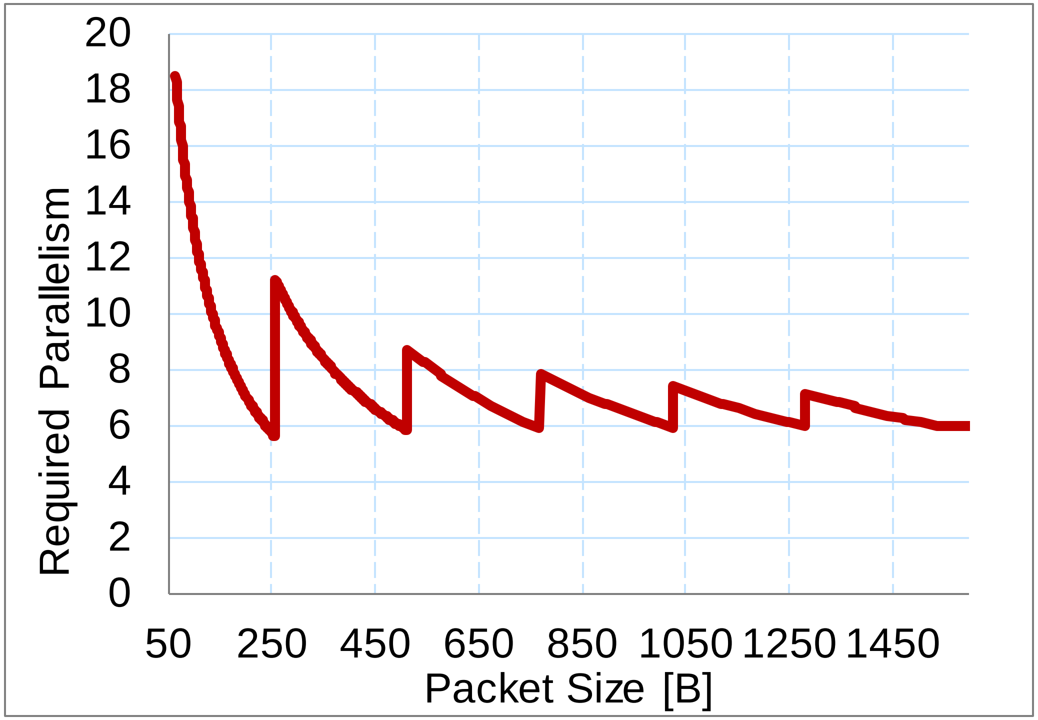
12.8Tbps Switches!

Lets convert this to packet rate requirements:

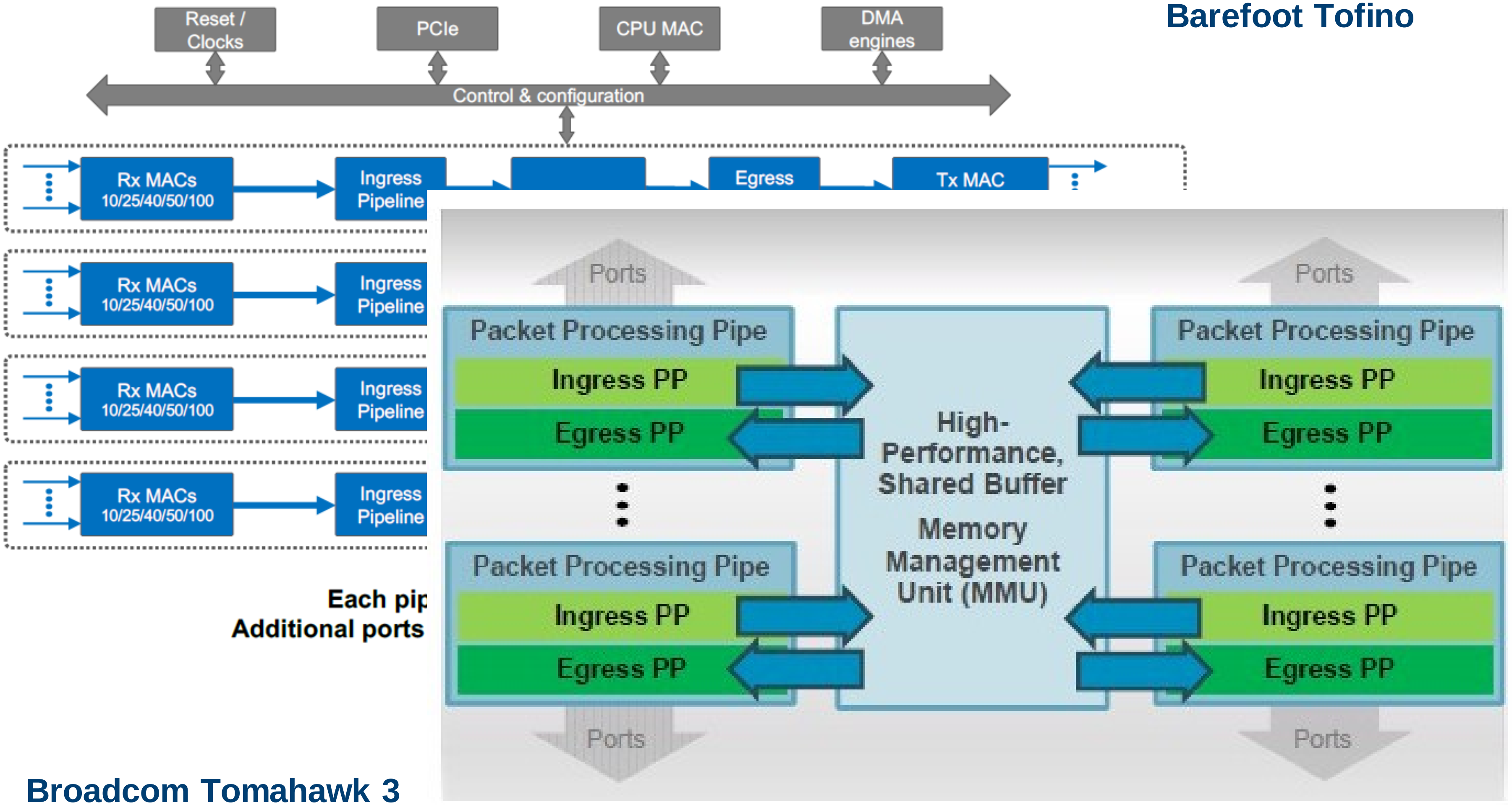
5800 Mpps @ 256B

19048 Mpps @ 64B

But clock rate is only ~1GHz….



# Multi-Core Switch Design



Barefoot Tofino

Broadcom Tomahawk 3

Image sources: https://p4.org/assets/p4_d2_2017_programmable_data_plane_at_terabit_speeds.pdf
https://www.nextplatform.com/2018/01/20/flattening-networks-budgets-400g-ethernet/

# Multi Core Switch Design

- So what? Multi-core in CPUs for over a decade

- Network devices are not like CPUs:

  – CPU: Pipeline - instructions, memory – data

  –Switch: pipeline – data, memory – control

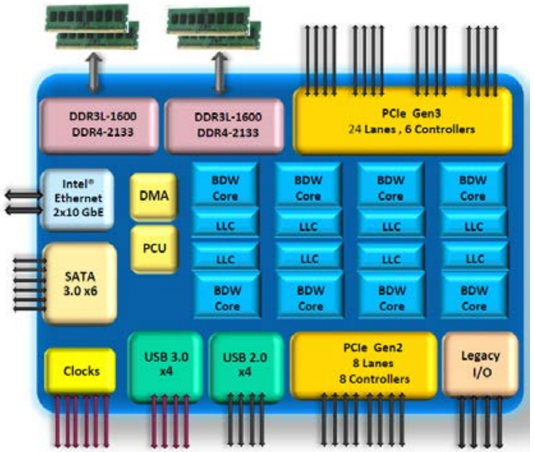
- Network devices have a strong notion of *time*

  –*Must* process the header on cycle X

  –Headers are split across clock cycles

  –Pipelining is the way to achieve performance

# Multi Core Switch Design



- The limitations of processing packets in the host:
- DPDK is a popular set of **libraries and drivers** for fast packet processing
- DPDK can process a packet in 80 clock cycles
  - Lets assume 4GHz clock (0.25ns/cycle)
  - Can process $4 \times 10^9 \div 80 = 50 \times 10^6$ pkts/sec
  - 50Mpps is not sufficient for 40GE. 30% of 64B packets at 100GE.
  - Can dedicate multiple cores…
  - And this is just sending / receiving, not operating on the packet!

# Multi Core Switch Design

- The problem with multi-core switch design: look up tables.
  - Shared tables:
    - need to allow access from multiple pipelines
    - need to support query rate at packet rate
  - Separate tables:
    - wastes resources
    - need to maintain consistency
      - Not everyone agrees with this assumption

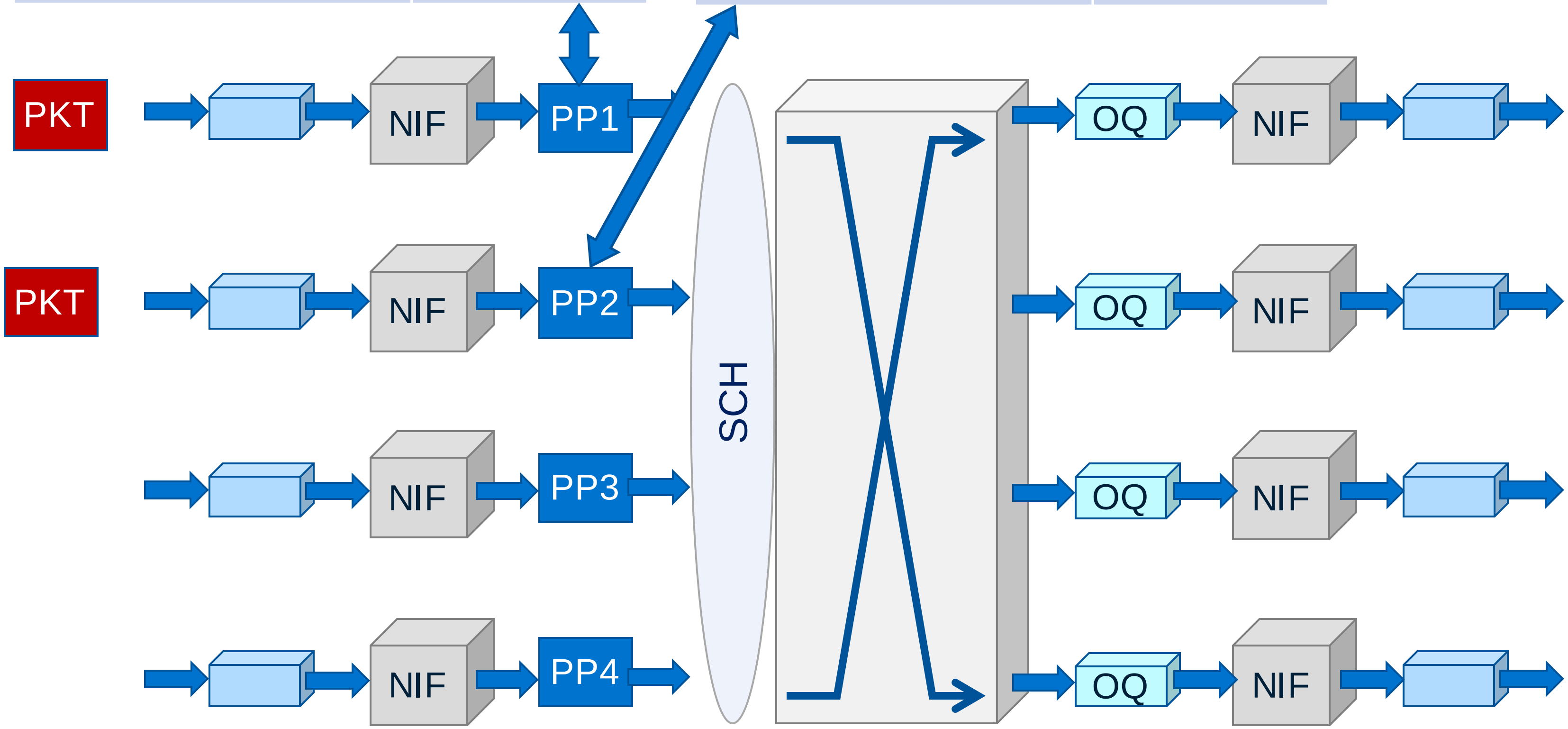
# Multi Core Switch Design

Table PP1

| DST MAC | DST Port |
|---|---|
| aa:bb:cc:dd:ee:ff | **3** |

Table PP2

| DST MAC | DST Port |
|---|---|
| aa:bb:cc:dd:ee:ff | **3** |



# High Throughput Interfaces

# Performance Limitations

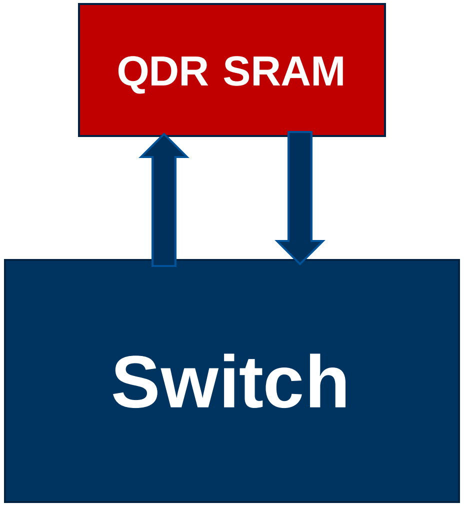
- So far we discussed performance limitations due to:
    - Data path
    - Network Interfaces
- Other common critical paths include:
    - Memory interfaces
        - Lookup tables, packet buffers
    - Host interfaces
        - PCIe, DMA engine

# Memory Interfaces

- On chip memories
  - Advantage: fast access time
  - Disadvantage: limited size (10’s of MB)
- Off chip memory:
  - Advantage: large size (up to many GB)
  - Disadvantage: access time, cost, area, power
- New technologies
  - Offer mid-way solutions

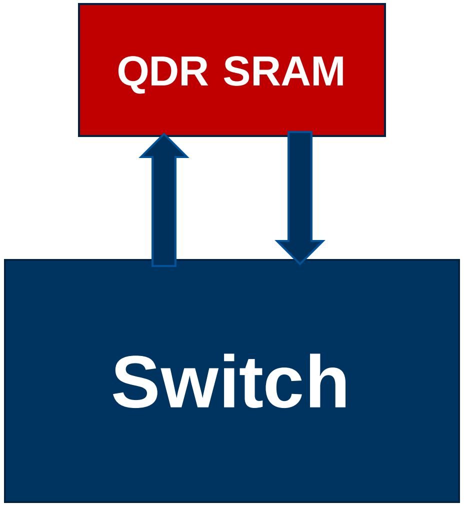
# Example: QDR-IV SRAM

- Does 4 operations every clock: 2 READs, 2 WRITEs
- Constant latency
- Maximum random transaction rate: 2132 MT/s
- Maximum bandwidth: 153.3Gbps
- Maximum density: 144Mb
- Example applications: Statistics, head-tail cache, descriptors lists



# Example: QDR-IV SRAM

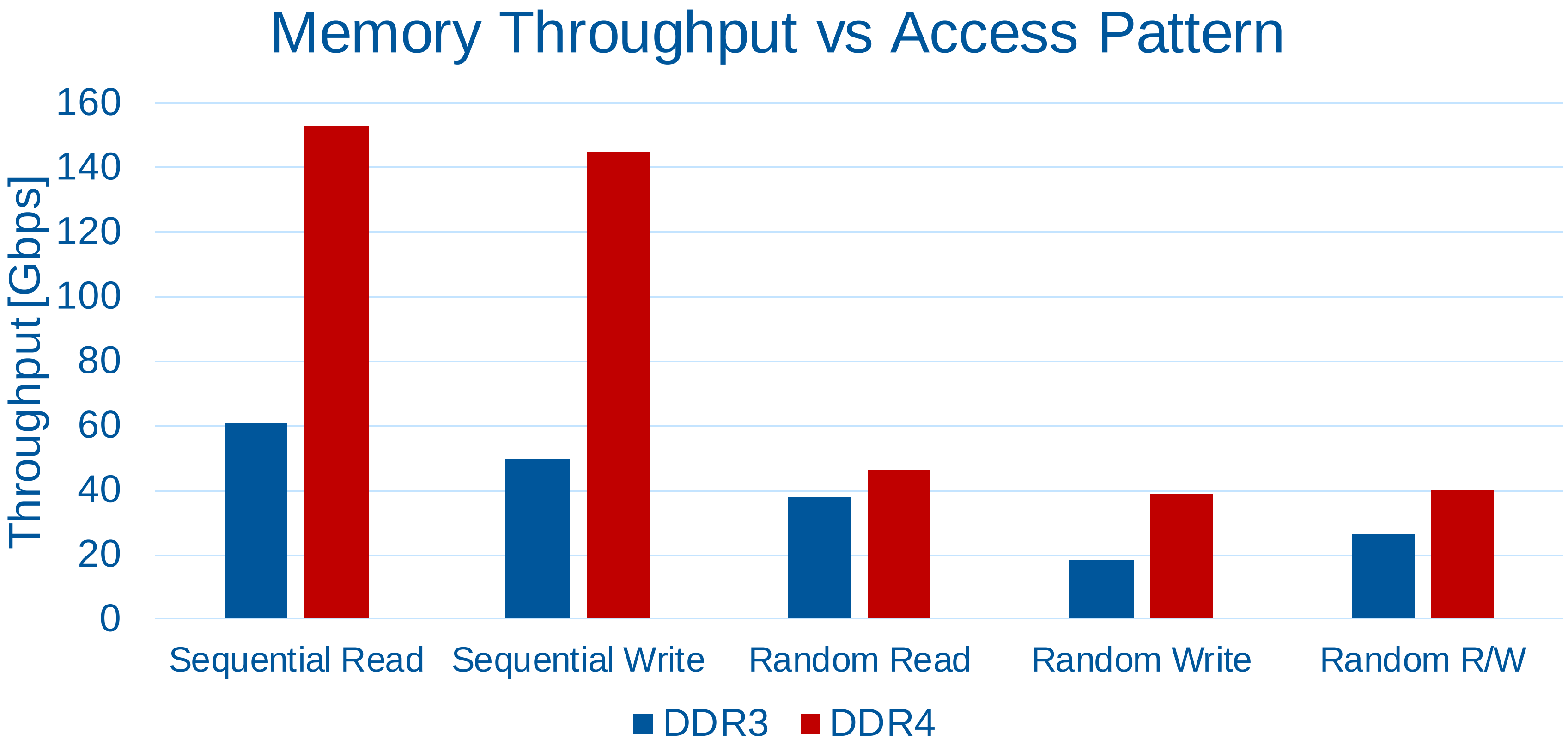
- Does 4 operations every clock: 2 READs, 2 WRITEs
  - *DDR4 DRAM: 2 operations every clock*
- Constant latency
  - *DDR4 DRAM: variable latency*
- Maximum random transaction rate: 2132 MT/s
  - DDR4 DRAM: 20MT/s (worst case! $t_{RC}$~50ns)
    - DDR4 theoretical best case 3200MT/s
- Maximum bandwidth: 153.3Gbps
  - DDR4 DRAM maximum bandwidth: 102.4Gbps (for 32b (2x16) bus)
- Maximum density: 144Mb
  - **DDR4 maximum density: 16Gb**
- Example applications: Statistics, head-tail cache, descriptors lists
  - No longer applicable: packet buffer



# Random Memory Access

- Random access is a “killer” when accessing DRAM based memories
  - Due to strong timing constraints
- Examples: rules access, packet buffer access
- DRAMs perform well (better) when there is strong locality or when accessing large chunks of data
  - E.g. large cache lines, files etc.
  - Large enough to hide timing constraints
    - E.g. for 3200MT/s, 64b bus: 50ns~ 1KB

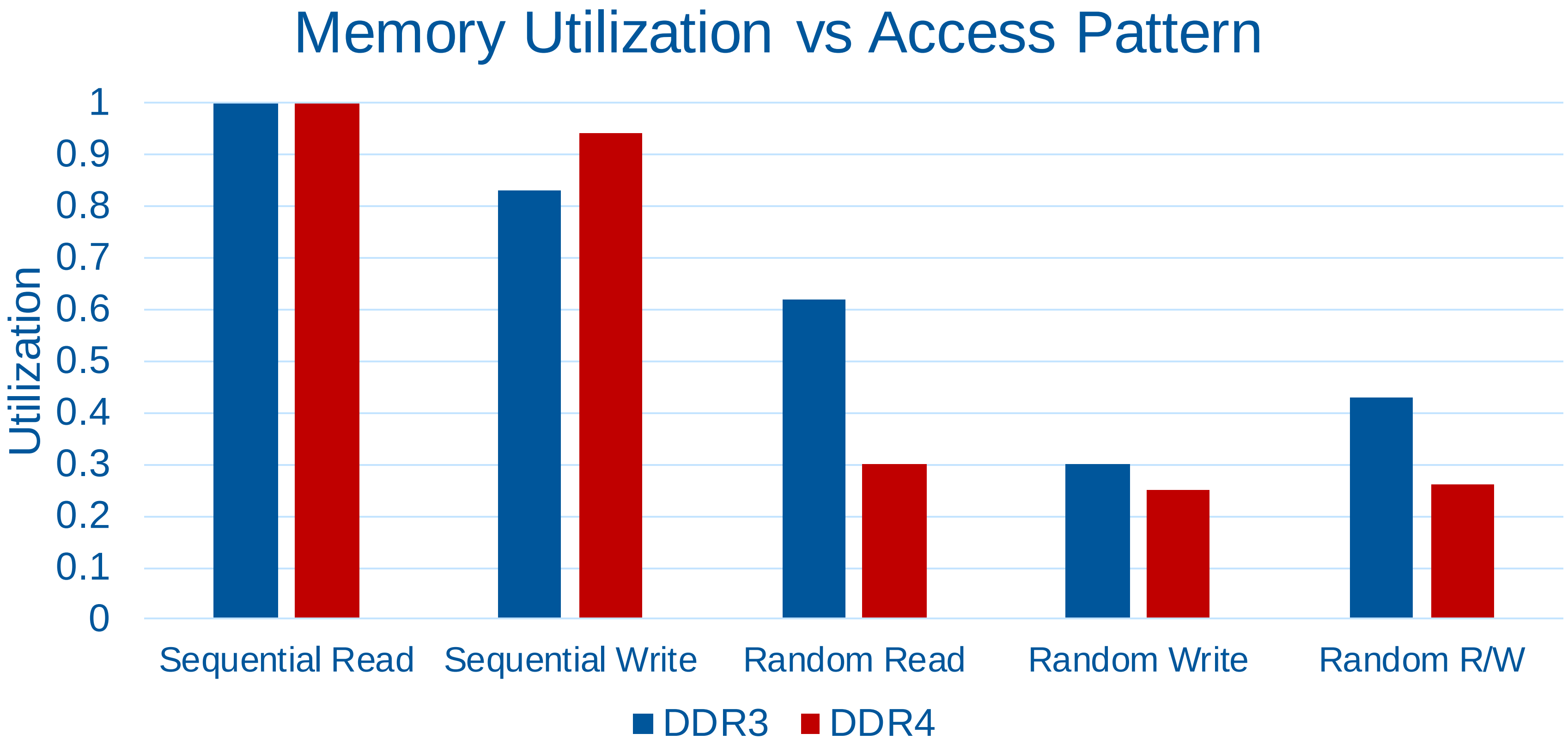
# Memory Access Pattern vs Performance



DDR3 on NetFPGA SUME, 1600MT/s

DDR4 on VCU1525, 2400MT/s

# Memory Access Pattern vs Performance

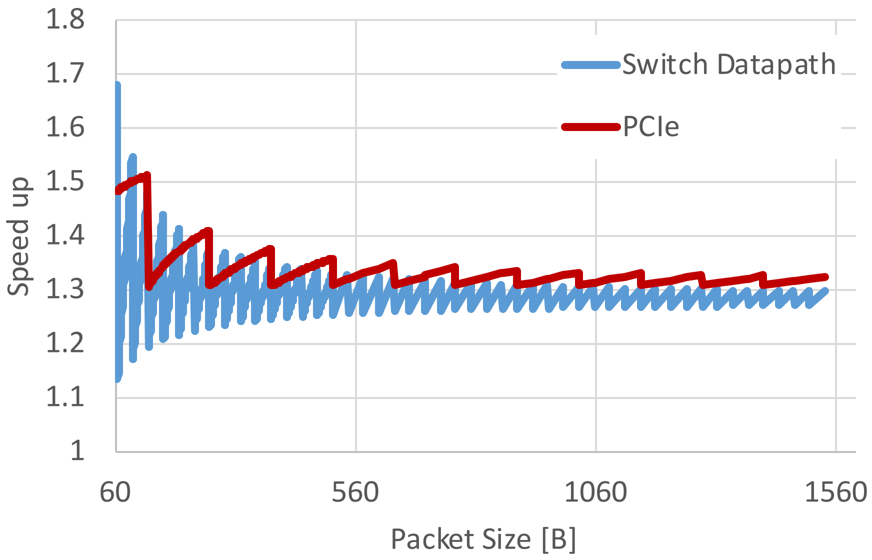


DDR3 on NetFPGA SUME, 1600MT/s
DDR4 on VCU1525, 2400MT/s

UNIVERSITY OF CAMBRIDGE

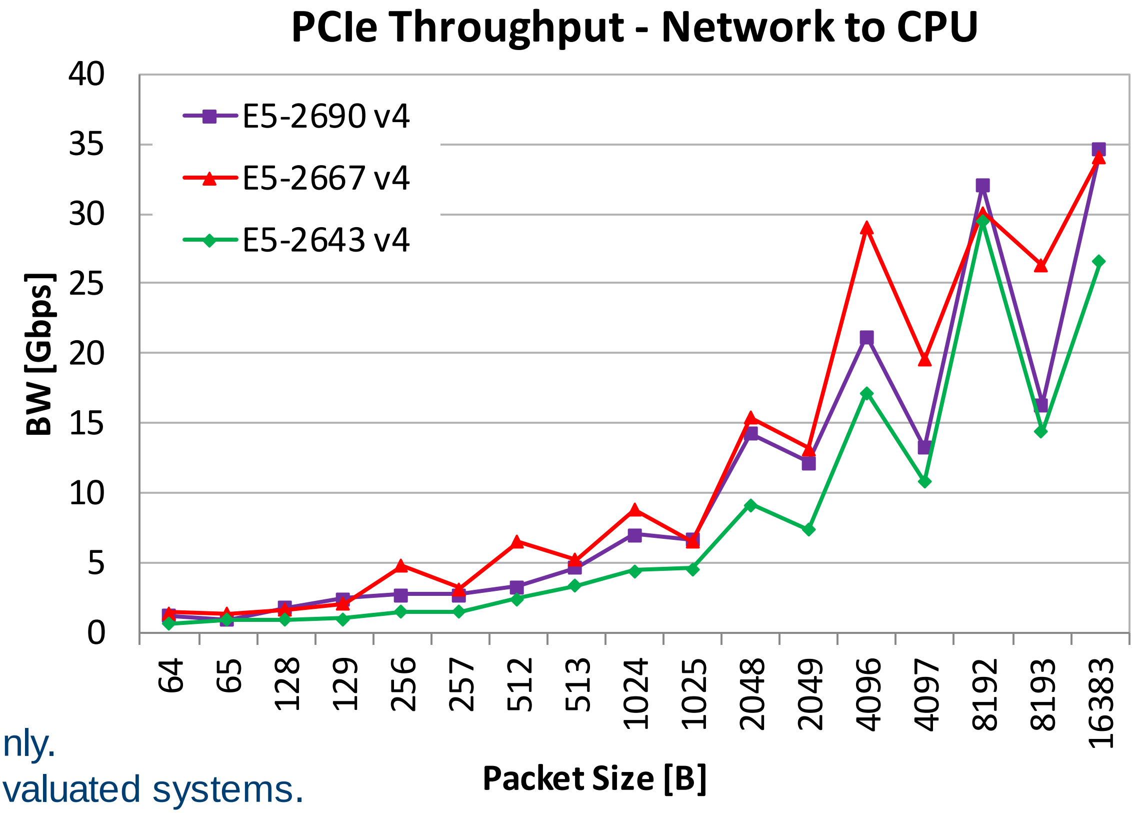
# Example: PCI Express Gen 3, x8

- The theoretical performance profile:
- PCIe Gen 3 – each lane runs at 8Gbps
- ~97% link utilization (128/130 coding, control overheads)
- Data overhead – 24B-28B (including headers and CRC)
- Configurable MTU (e.g., 128B, 256B, …)



# Example: PCI Express Gen 3, x8

- Actual throughput on VC709, using Xilinx reference project: (same FPGA as NetFPGA SUME)
- This is so far from the performance profile…
- Why?



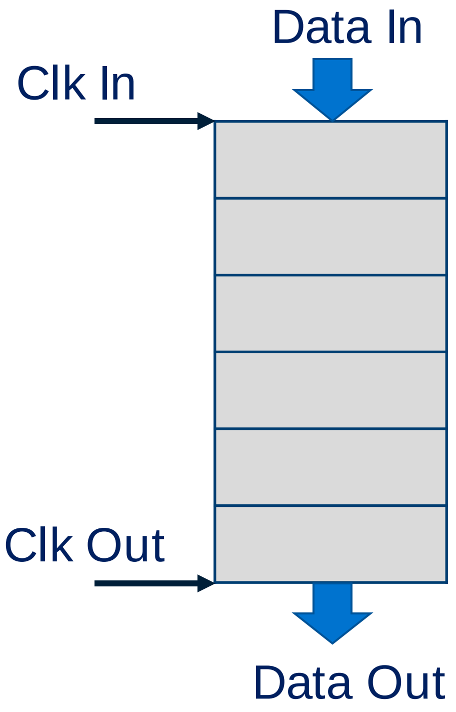
Note: the graph is for illustration purposes only.
There were slight differences between the evaluated systems.

UNIVERSITY OF CAMBRIDGE

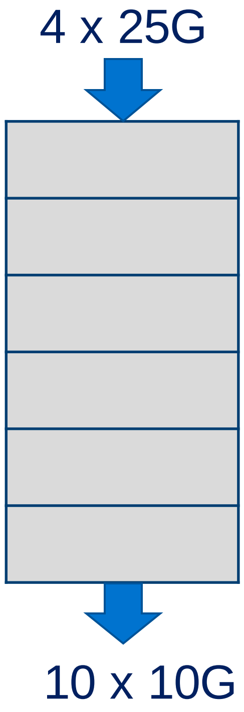
# Flow Control

# Crossing Clock Domains

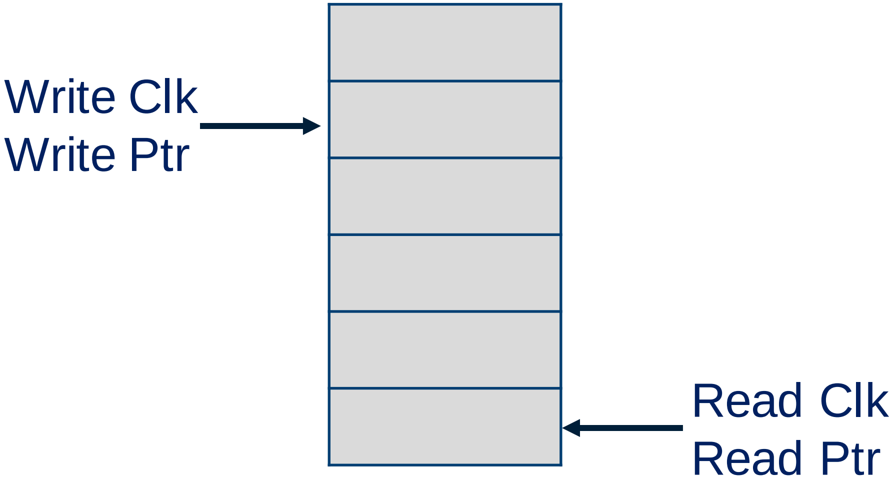
- Last week we discussed the clock frequency required in different places in the design.
- Crossing clock domains requires careful handling



**Asynchronous FIFO**



**Gear Box**



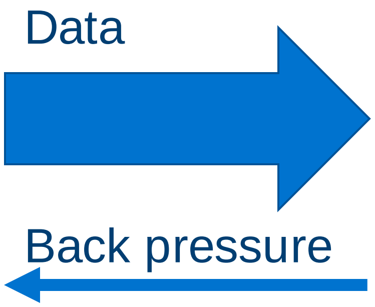
**Synchronizer**

# Crossing Clock Domains

- Why do we care about clock domain crossing?
- Adds latency
- The latency is not deterministic
    - But bounded
- Crossing clock domains multiple times increases the jitter
- Using a single clock is often not an option:
    - Insufficient packet processing rate
    - Multiple interface clocks
    - Need speed up (e.g., to handle control events)

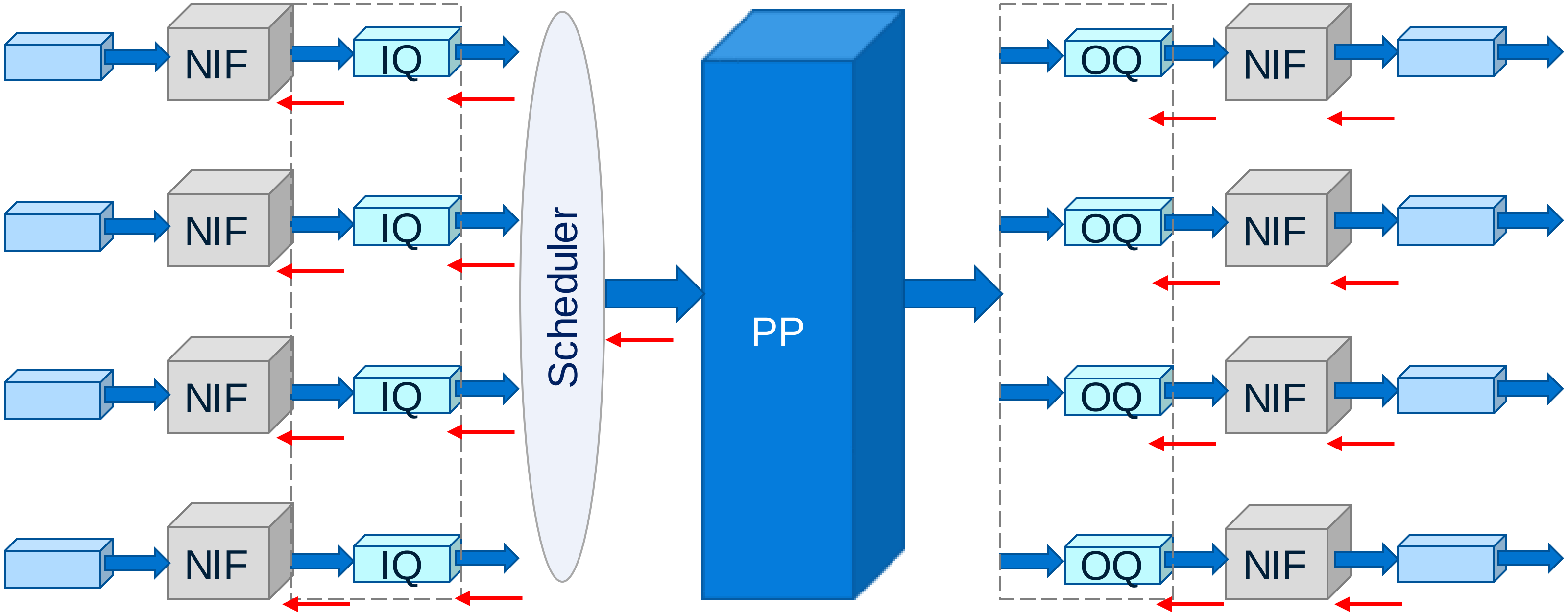
# Flow Control

- The flow of the data through the device (the network) needs to be regulated
- Different events may lead to stopping the data:
  - An indication from the destination to stop
  - Congestion (e.g. 2 ports sending to 1 port)
  - Crossing clock domains
  - Rate control
  - …



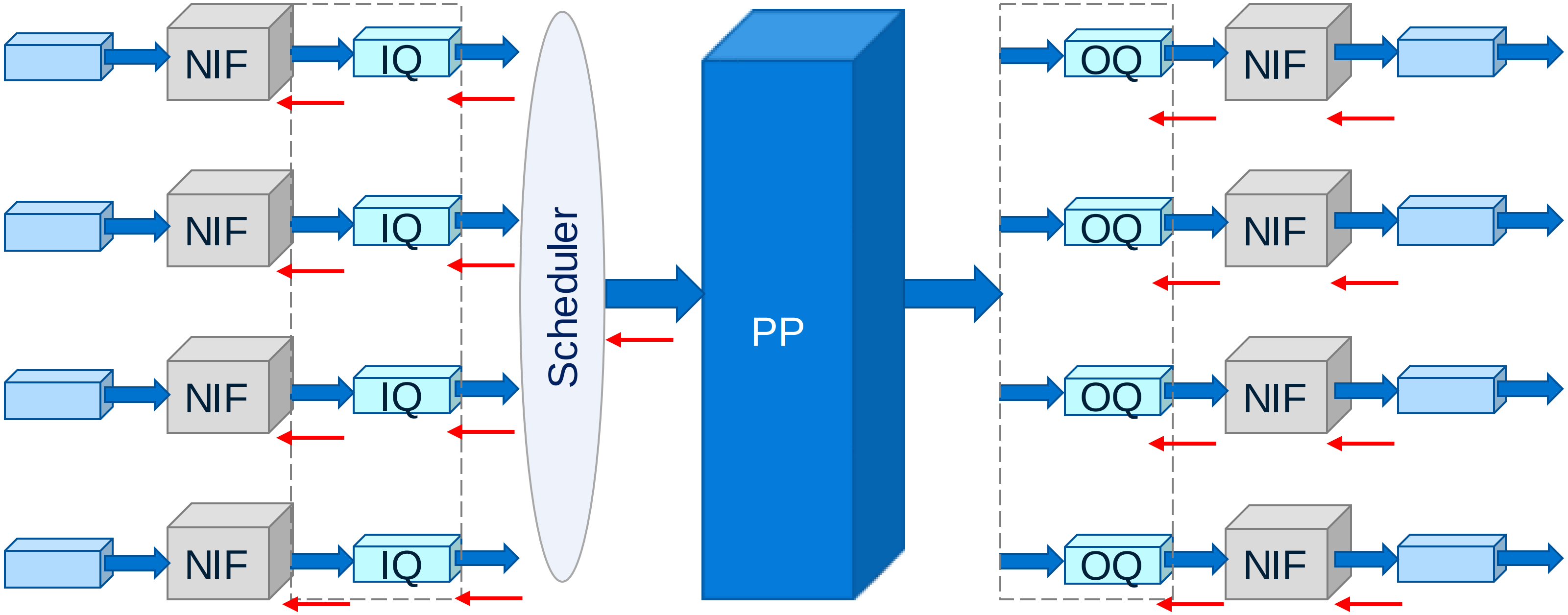
# Flow Control

- Providing back pressure is not always allowed
- In such cases, need to make amendments in the design



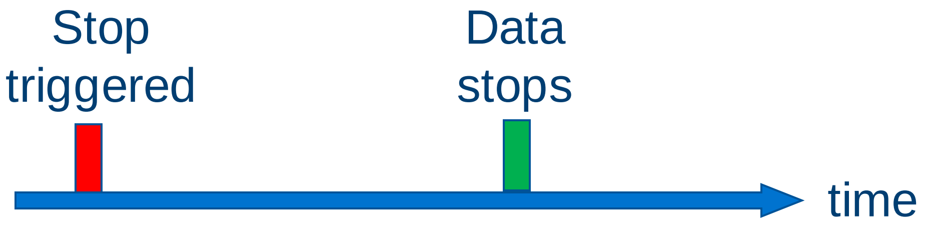
# Flow Control

- What to do if an output queue is congested?



# Flow Control and Buffering

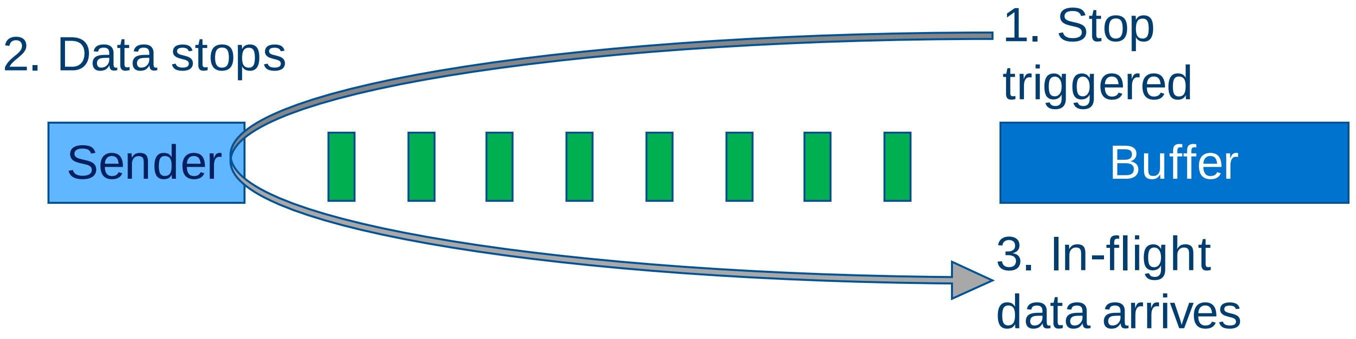
- Back pressure may take *time*



- Need to either:
  - Assert back pressure sufficient time before traffic needs to stop OR
  - Provide sufficient buffering

# Flow Control and Buffering

Calculating buffer size:



Intuitively:

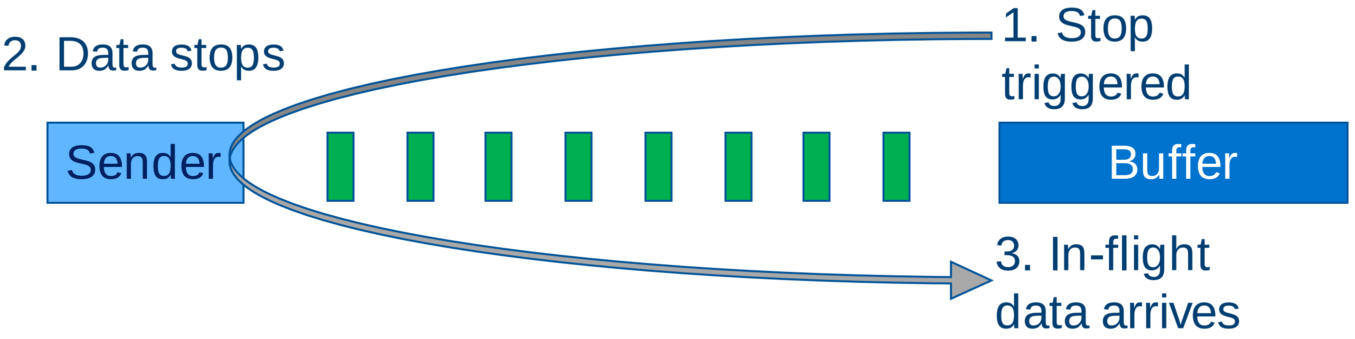
Nearby sender: Buffer size ≥ Reaction time × Data rate

Remote sender: Buffer size ≥ RTT × Data rate

Buffer size ≥ (RTT + Reaction time) × Data rate

# Flow Control and Buffering

Calculating buffer size:



2 switches, connected using 100m fibre, 10G port, instantaneous response time:

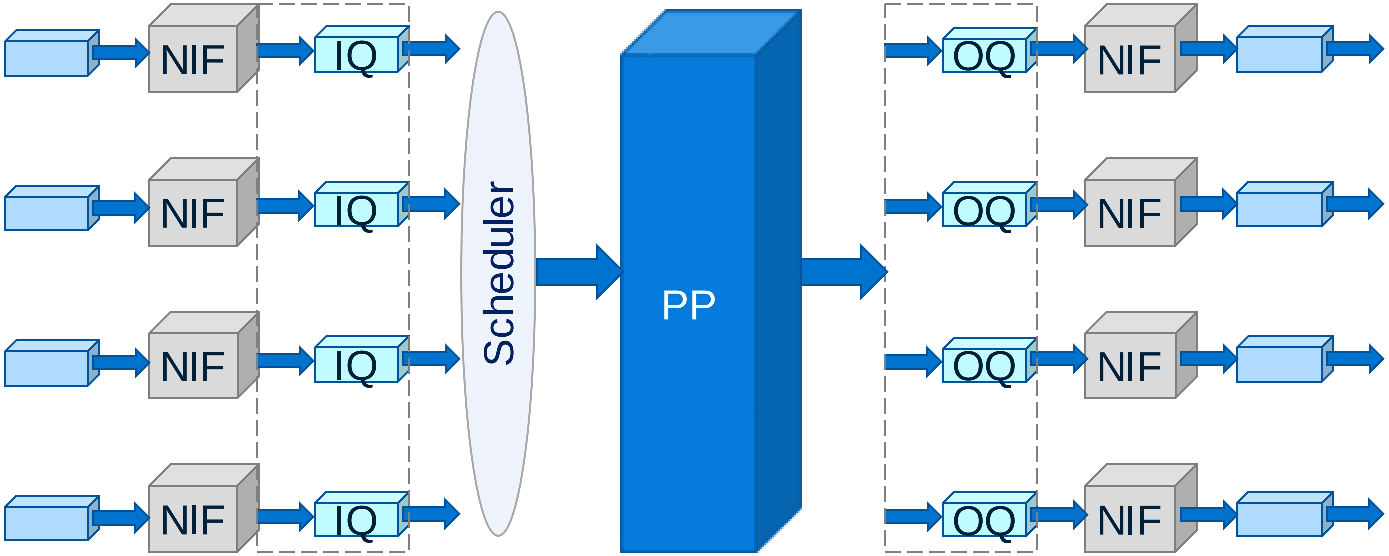
Propagation delay in a fibre is 5ns/m

Buffer size ≥ 1us × 10Gbps = ~1.25KB

# Low Latency Switches

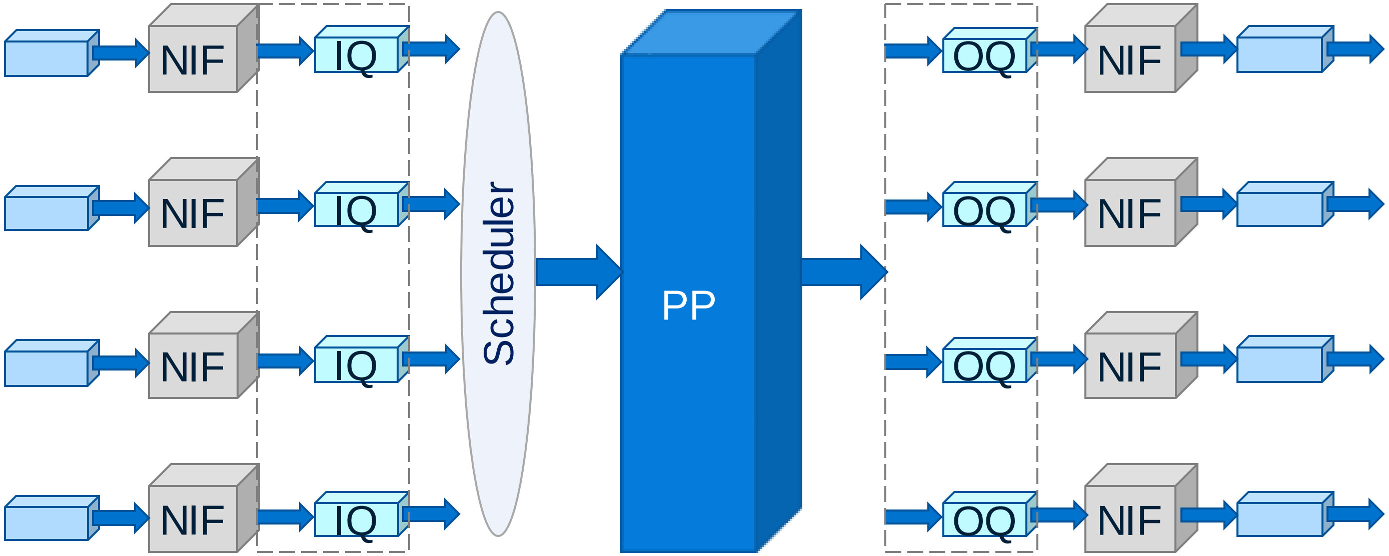
# How to lower the latency of a switch?

- Obvious option 1: Increase clock frequency
  - –E.g. change core clock frequency from 100MHz to 200MHz
  - –Half the time through the pipeline



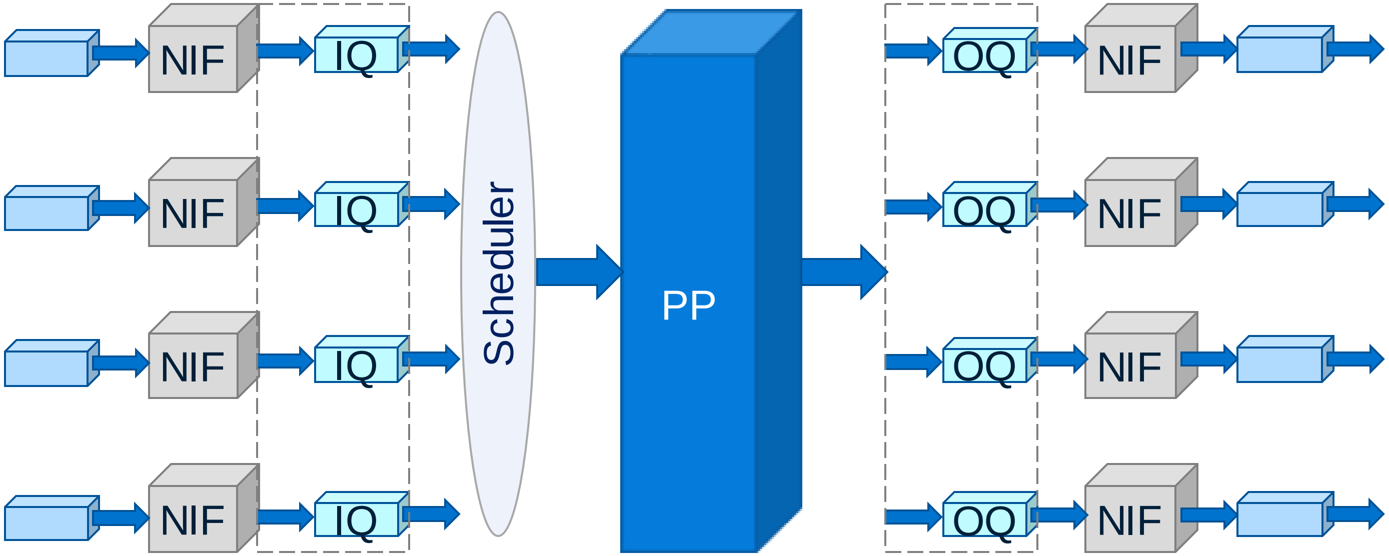
# How to lower the latency of a switch?

- Obvious option 1: Increase clock frequency
- Limitations:
  - Frequency is often a property of manufacturing process
  - Some modules (e.g. PCS) must work at a specific frequency (multiplications)



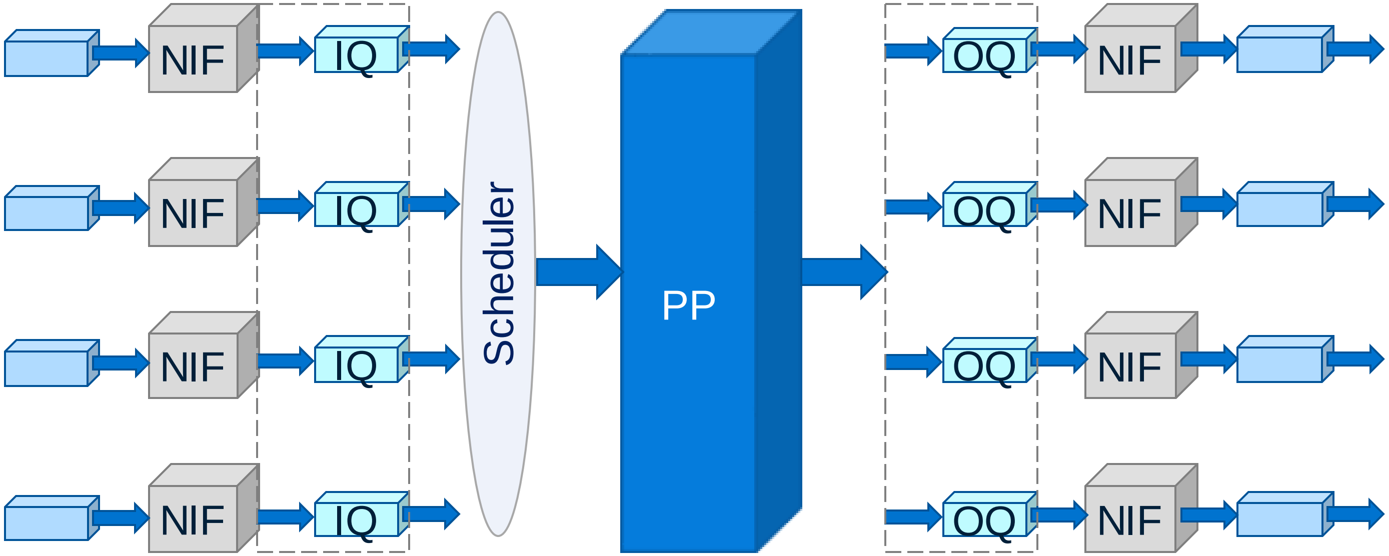
# How to lower the latency of a switch?

- Obvious option 2: Reduce the number of pipeline stages
  - –Can you do the same in 150 pipeline stages instead of 200?
  - –Limitation: hard to achieve.



# How to lower the latency of a switch?

- Can we achieve ~0 latency switch?
  - –Is there a lower bound on switch latency?


NIF
IQ
NIF
IQ
NIF
IQ
NIF
IQ
Scheduler
PP
OQ
NIF
OQ
NIF
OQ
NIF
OQ
NIF