Decompressing Snappy Compressed Files at the Speed of OpenCAPI

OpenPOWER Summit Europe

RAI Centre | Amsterdam

October 3-4, 2018

Speaker: **Jian Fang**

TU Delft

Quantum & Computer Engineering

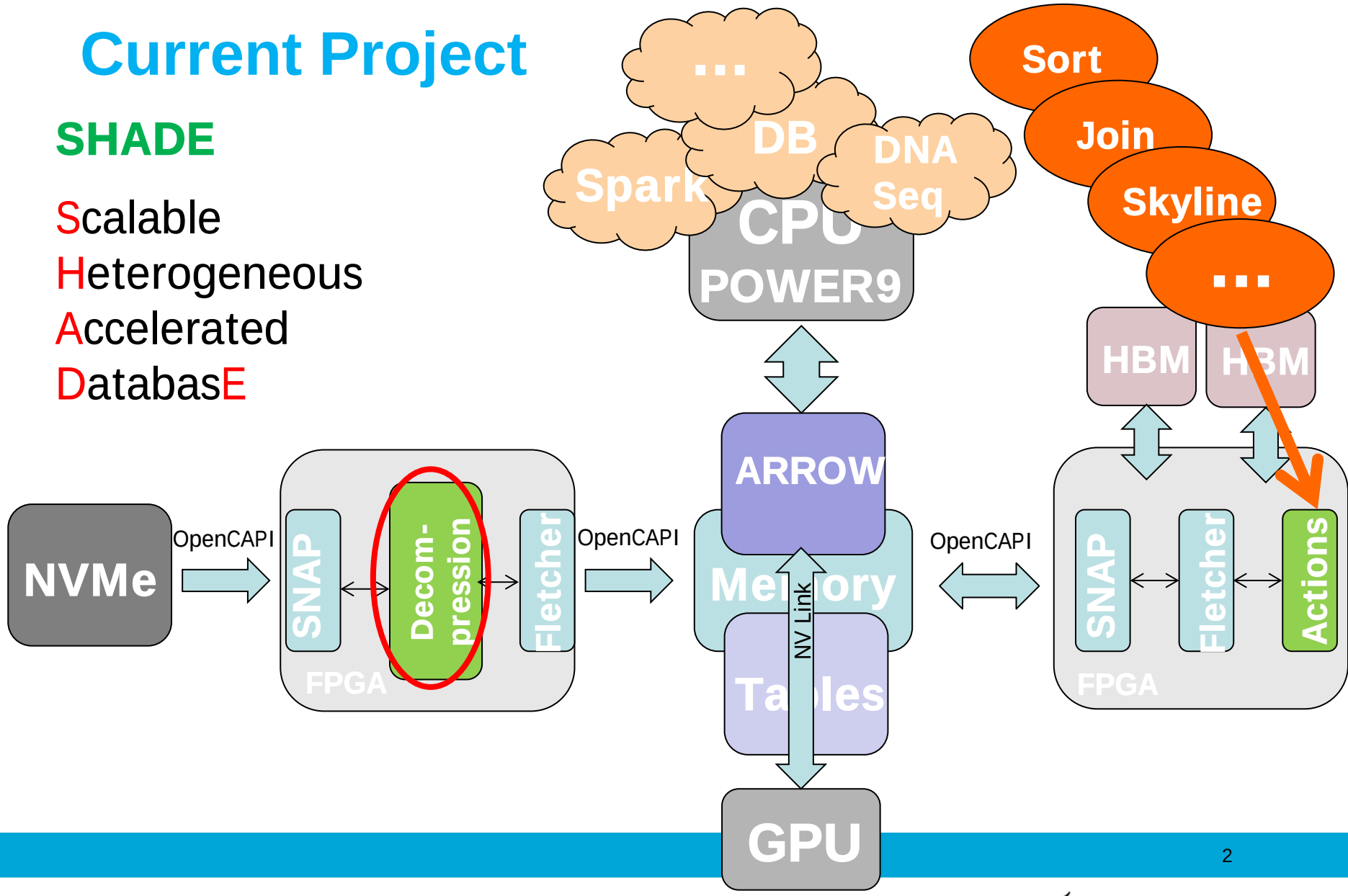
TU Delft

Delft University of Technology

Current Project

SHADE

Scalable
Heterogeneous
Accelerated
DatabasE



Quantum & Computer Engineering

TU Delft

Big Data Processing

Database Analytics

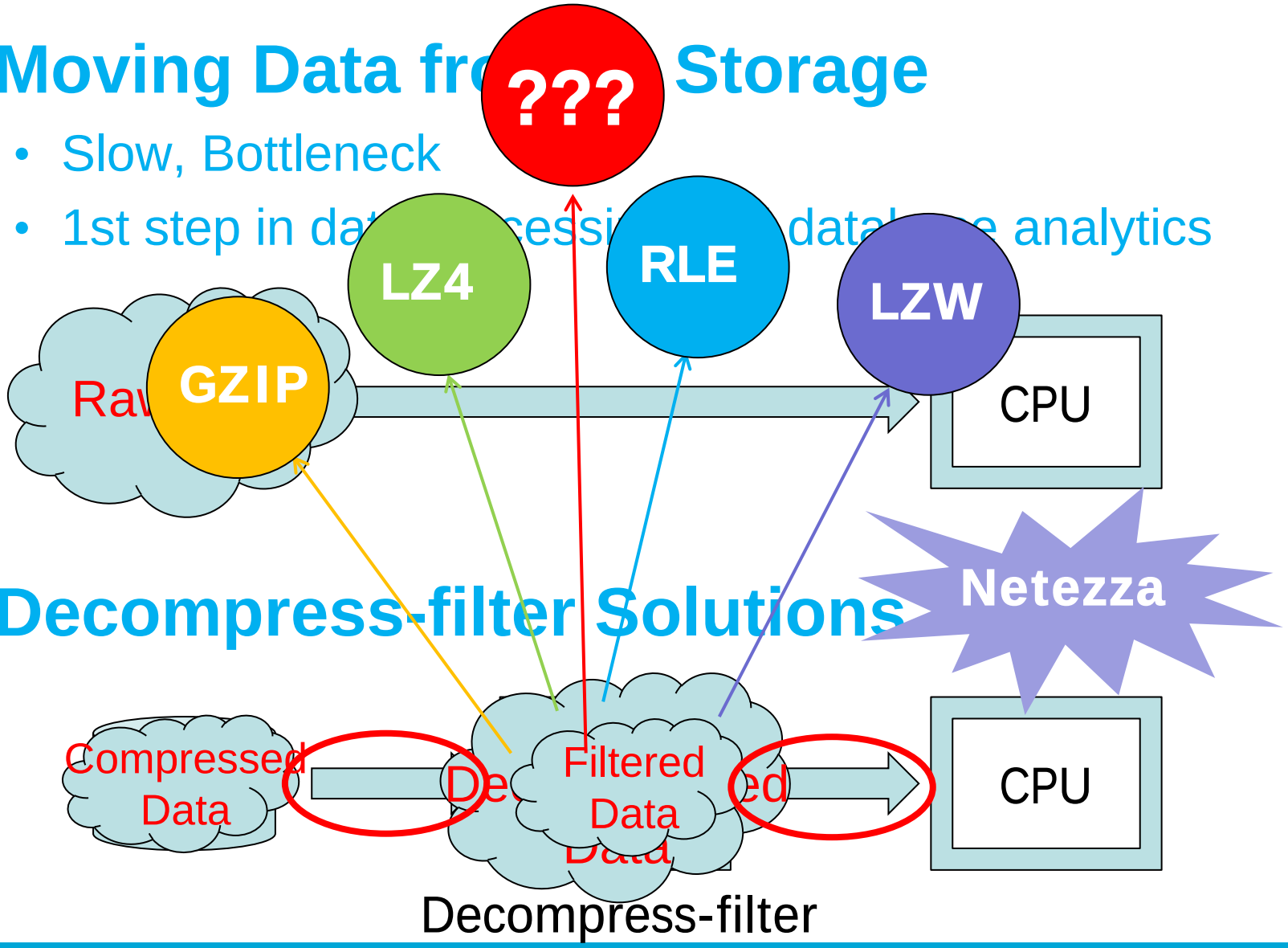
Sort Join Aggregation

Filter Grouping

Data reformat Aggregation

Data Movement

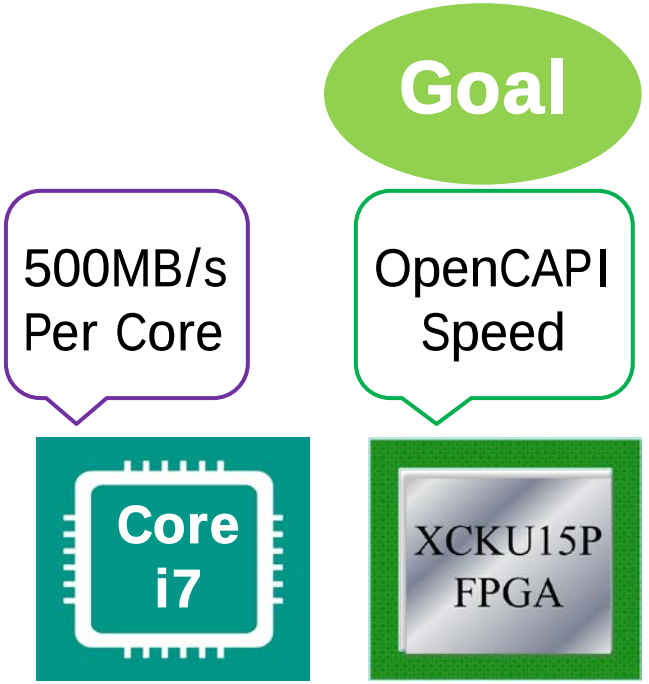
- **Moving Data fr Storage**
  - Slow, Bottleneck
  - 1st step in da cessi datal e analytics



- **Decompress-filter Solutions**

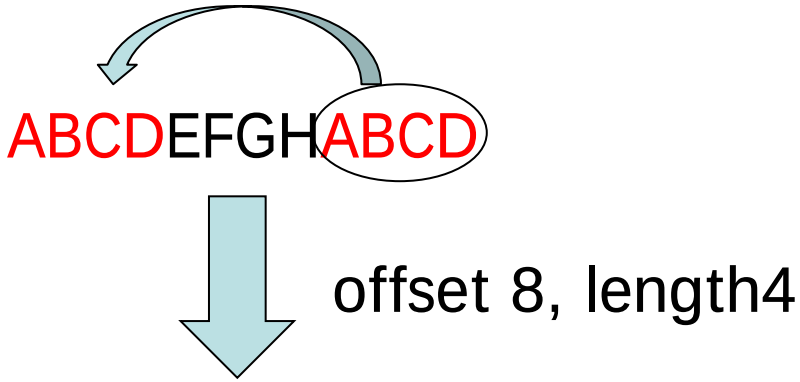
Snappy compression Algorithm

- LZ77-based, byte-level, lossless
- In Hadoop ecosystem
- Support Parquet, ORC, etc.
- Low compression ratio
- Fast compression and decompression



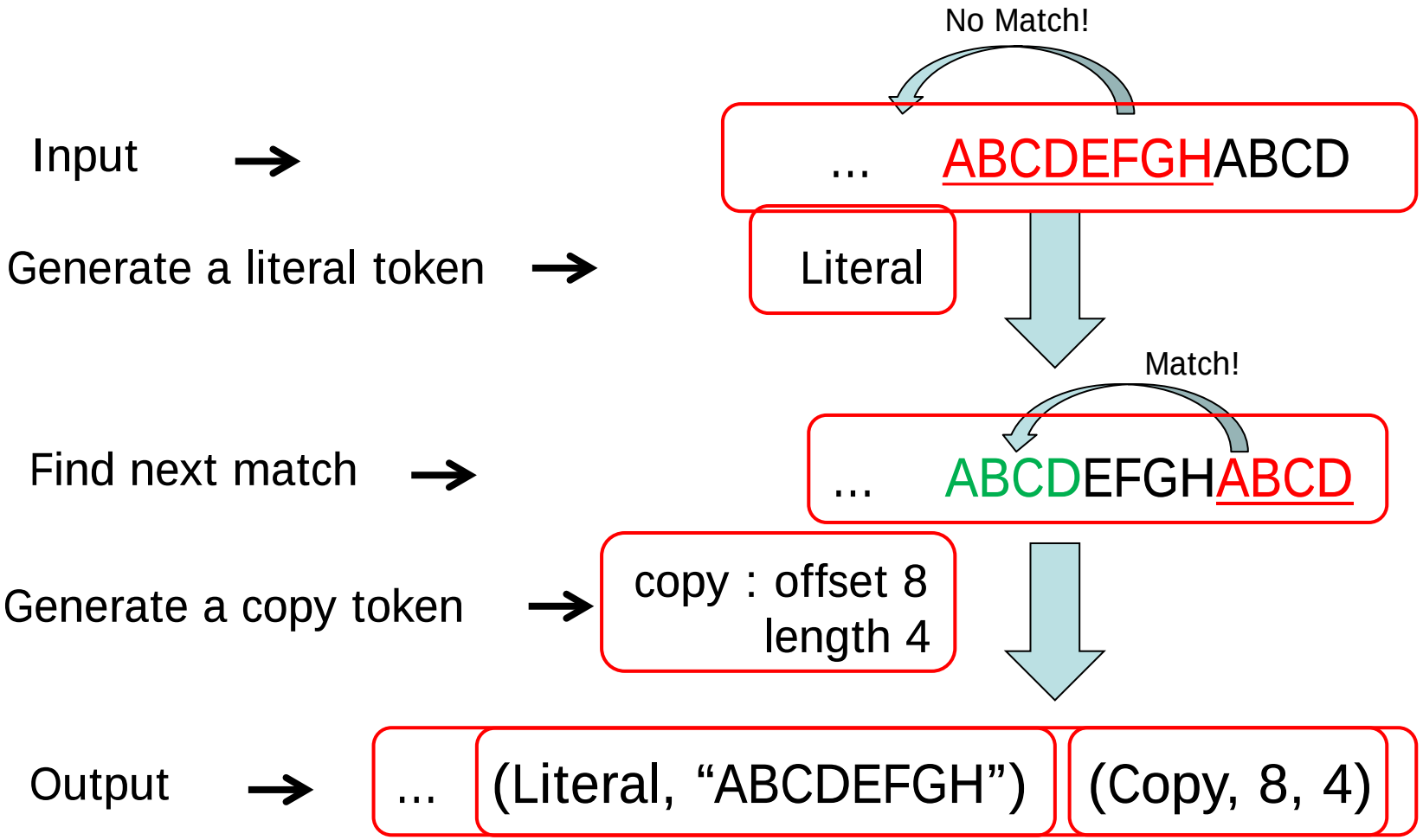
Mechanism of Snappy compression

- Two kinds of token: Literal token and copy token
- Find match from previous data
- Example:

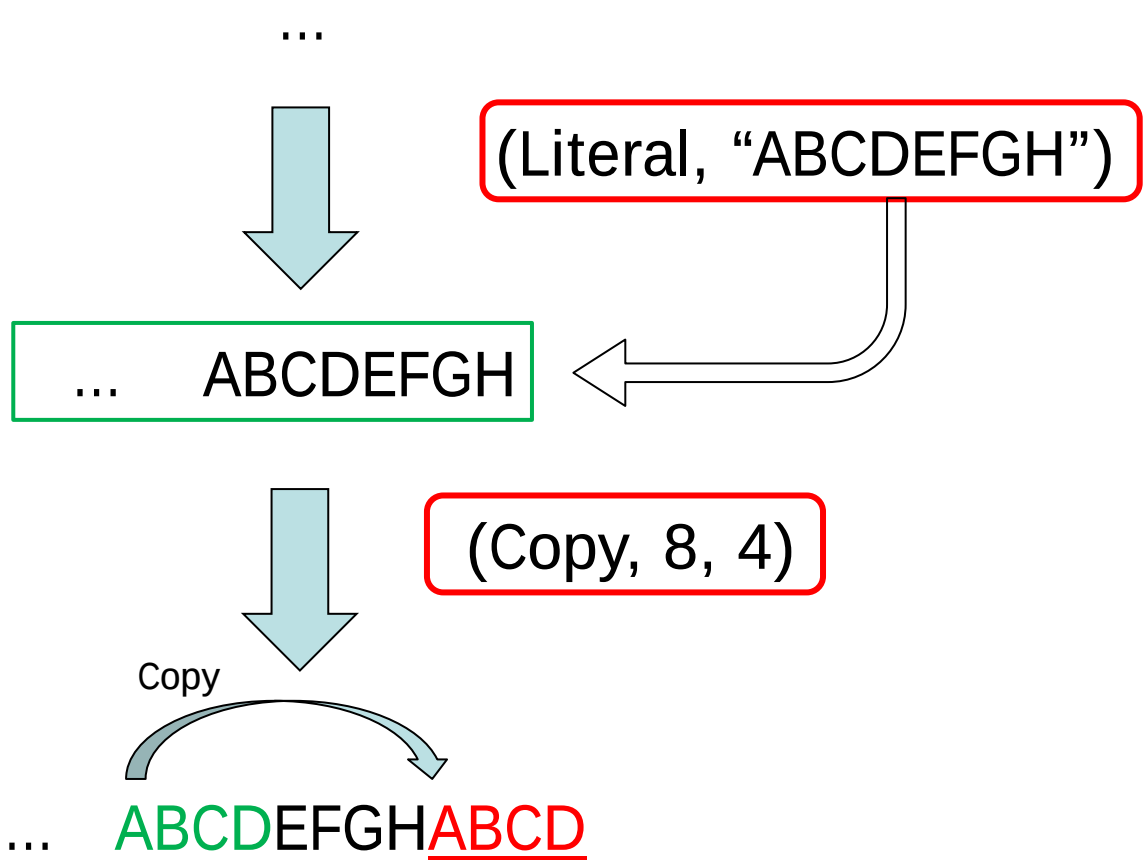


Use (8,4) to replace “ABCD”

Snappy Compression



Snappy Decompression



Snappy compression Algorithm

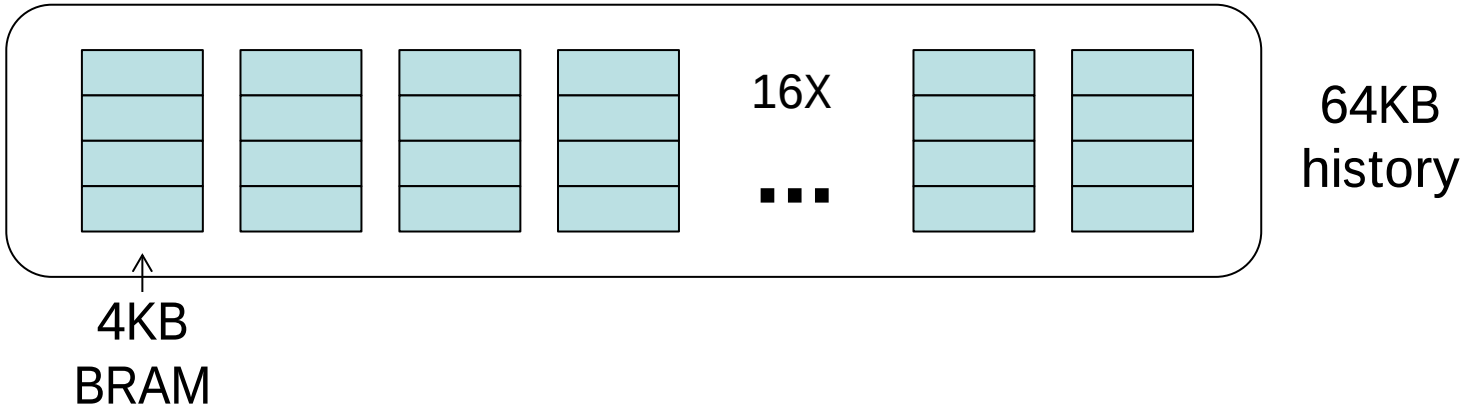
- An independent 64KB history for every 64KB data
- Token: copy or literal, copy lenth<=64B
- Token size: 2 Byte minimum, average ≈< 4 Byte(assumption for design)
- Highly data dependent

Benchmark	Compression Ratio	Average Token Size
DNA1	3.31	5.79
DNA2	2.36	7.38
DNA3	2.57	7.03
TPC-H	2.53	2.72
…	…	…

DNA data source: ftp://ftp.ncbi.nlm.nih.gov/ncbi-asn1

Design in FPGA

- An independent 64KB history for every 64KB data
- Token: copy or literal, copy lenth<=64B
- Token size: 2 Byte minimum, average ≈< 4 Byte(assumption for design)
- Highly data dependent



DNA data source: ftp://ftp.ncbi.nlm.nih.gov/ncbi-asn1

Design Challenge

- Deal with different types of tokens
- Handle various size of tokens
- Parallelize token translation
- Keep up with the interface bandwidth

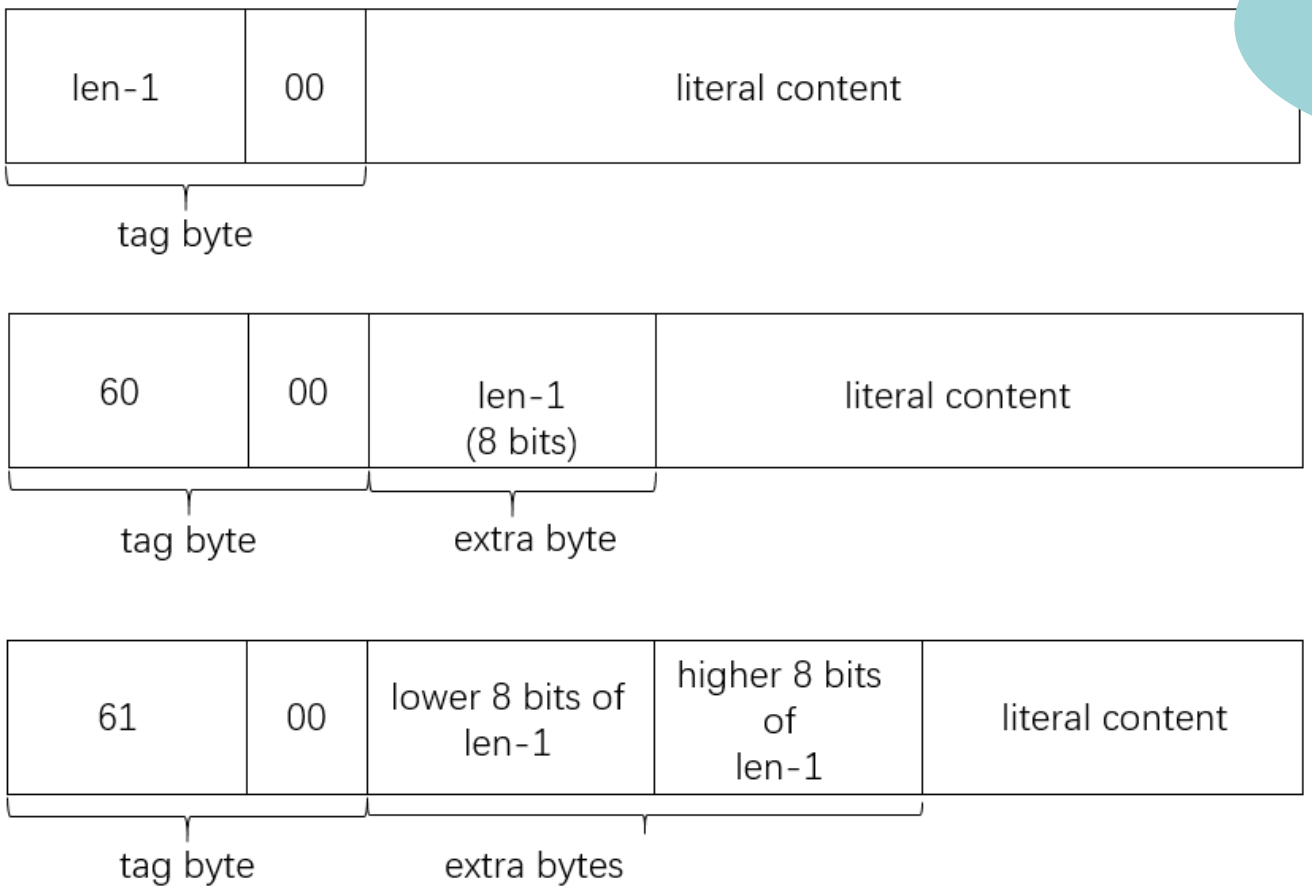
Design Challenge

- Deal with different types of tokens
- Handle various size of tokens
- Parallelize token translation
- Keep up with the interface bandwidth

Design Challenge

- Deal with different types of tokens
- Handle various size of tokens
- Parallelize token translation
- Keep up with the interface bandwidth

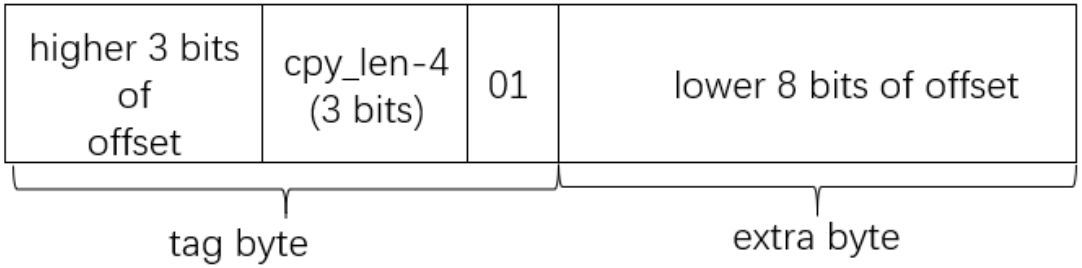
Token Structure



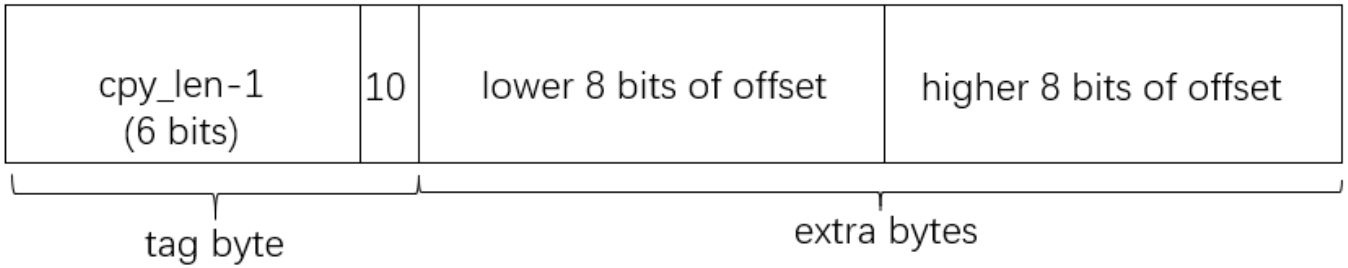
Literal Token

Source: Y. Qiao. An fpga-based snappy decompressor-filter. Master's thesis, Delft University of Technology, 2018.

Token Structure



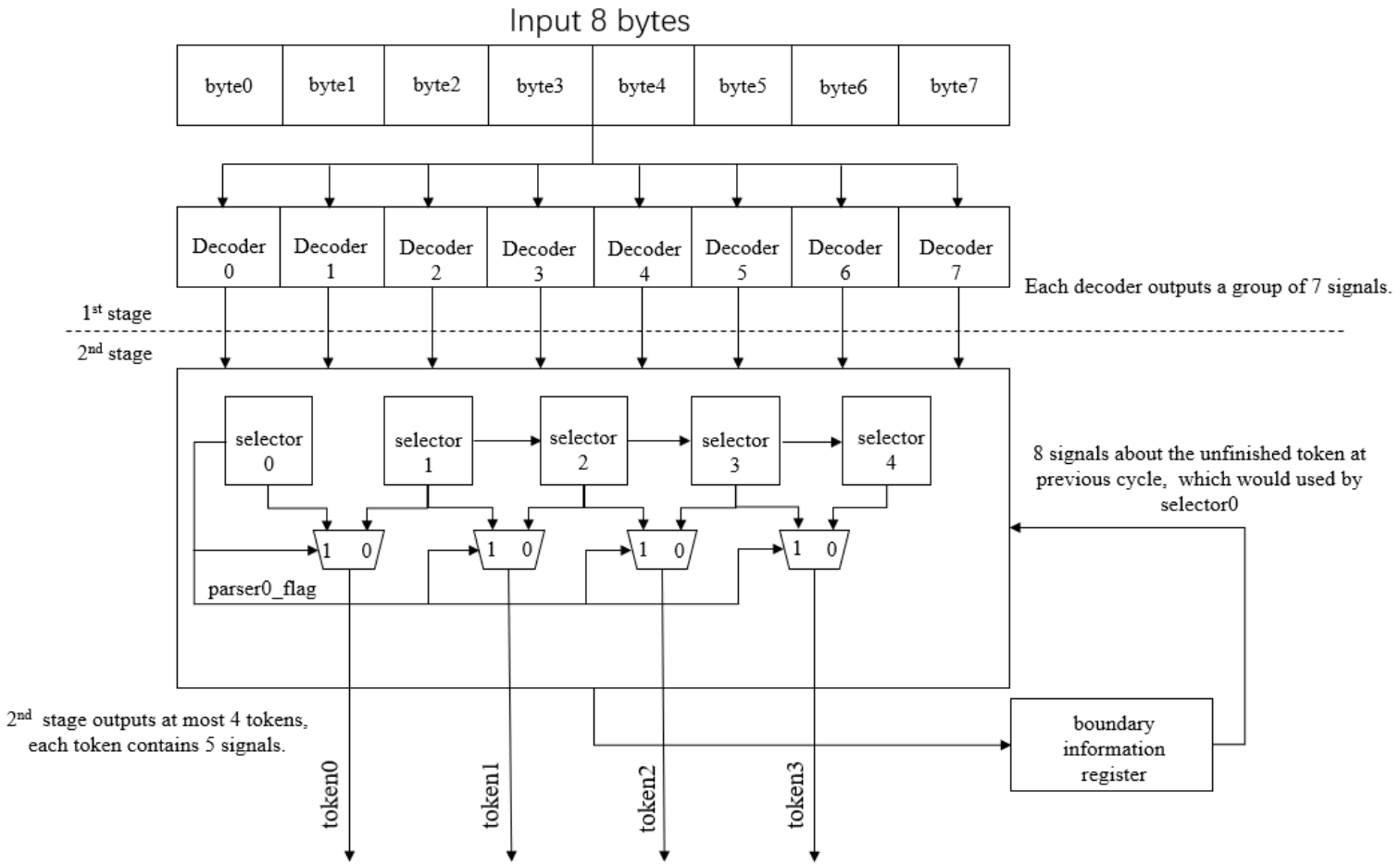
Copy Token



Tokens are in various length

Source: Y. Qiao. An fpga-based snappy decompressor-filter. Master's thesis, Delft University of Technology, 2018.

Prior Solutions: Decode all possiblities



Source: Y. Qiao. An fpga-based snappy decompressor-filter. Master's thesis, Delft University of Technology, 2018.

Quantum & Computer Engineering

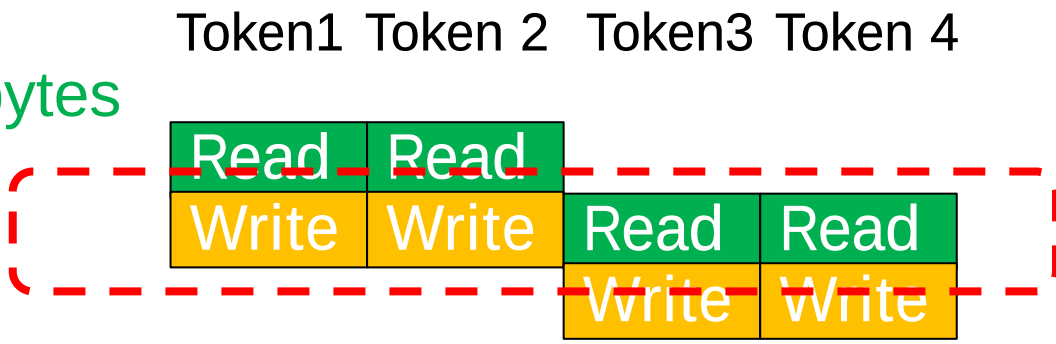
TU Delft

Design Challenge

- Deal with different types of tokens
- Handle various size of tokens
- Parallelize token translation
- Keep up with the interface bandwidth

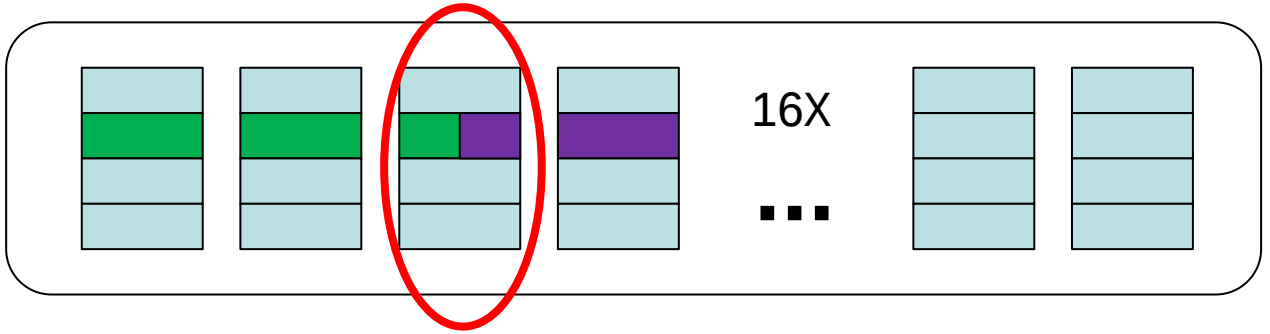
Token Dependency

- Read – Read
  - No dependency
- Write – Write
  - Never write same bytes
  - No dependency
- Write – Read
  - Forwarding



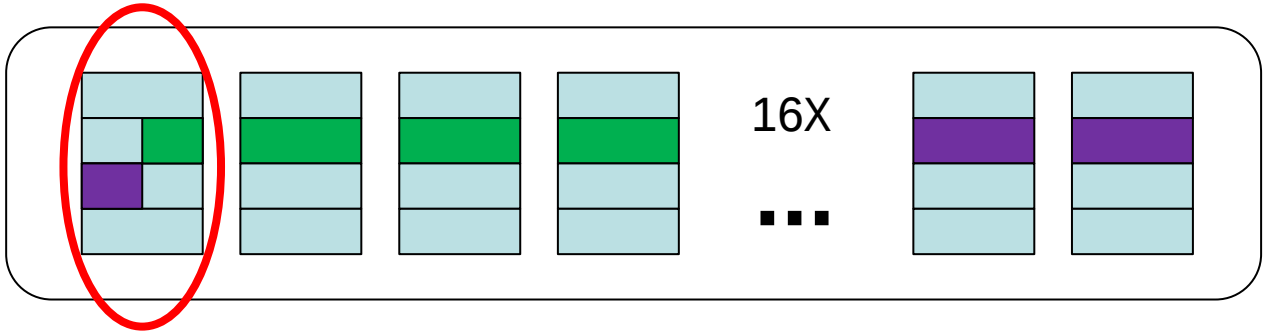
Token Dependency – Address Conflict

- Write – Write
  - Same block, same line



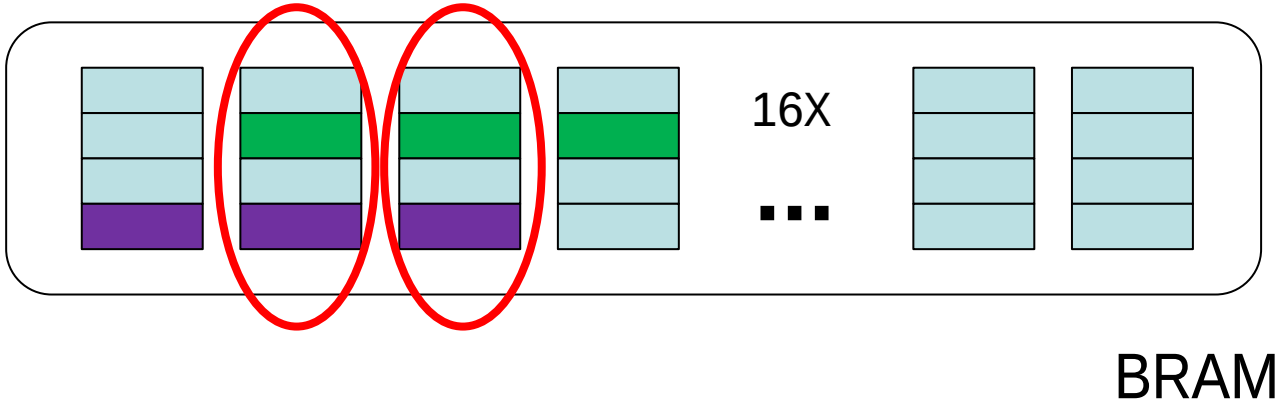
Token Dependency – Address Conflict

- Write – Write
  - Same block, different lines



Token Dependency – Address Conflict

- Read – Read



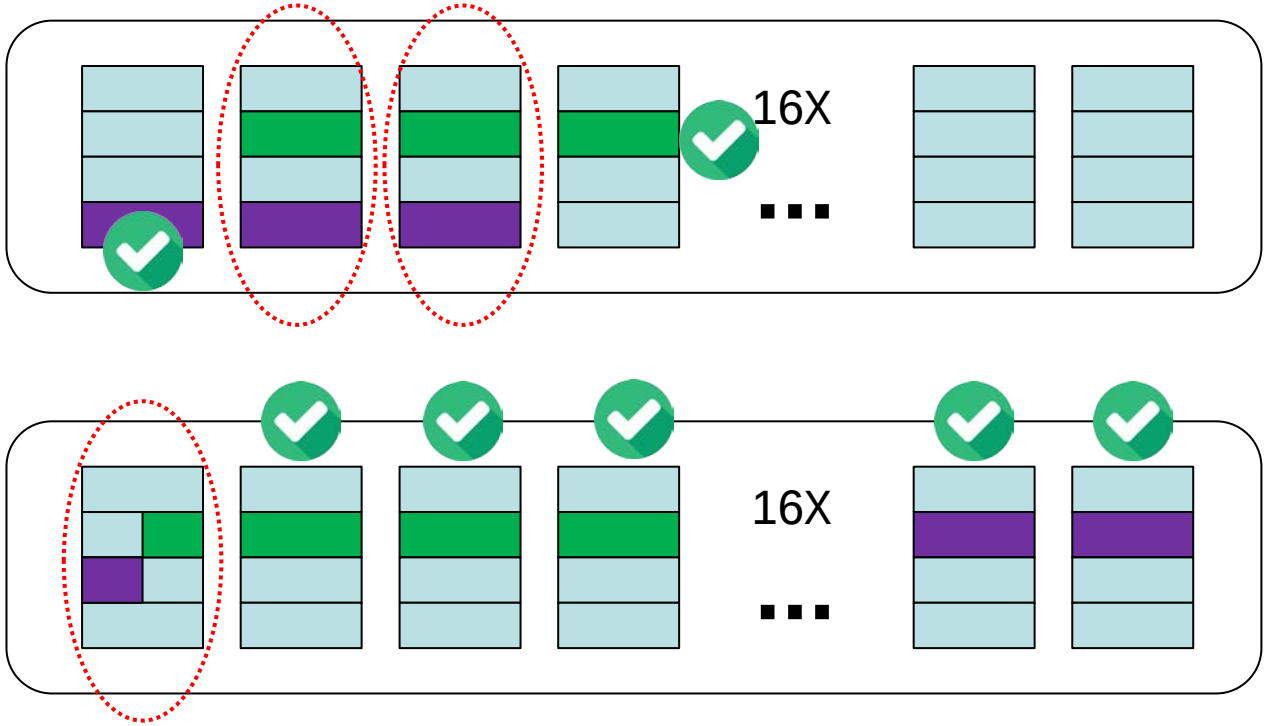
Design Challenge

- Deal with different types of tokens
- Handle various size of tokens
- Parallelize token translation
- Keep up with the interface bandwidth

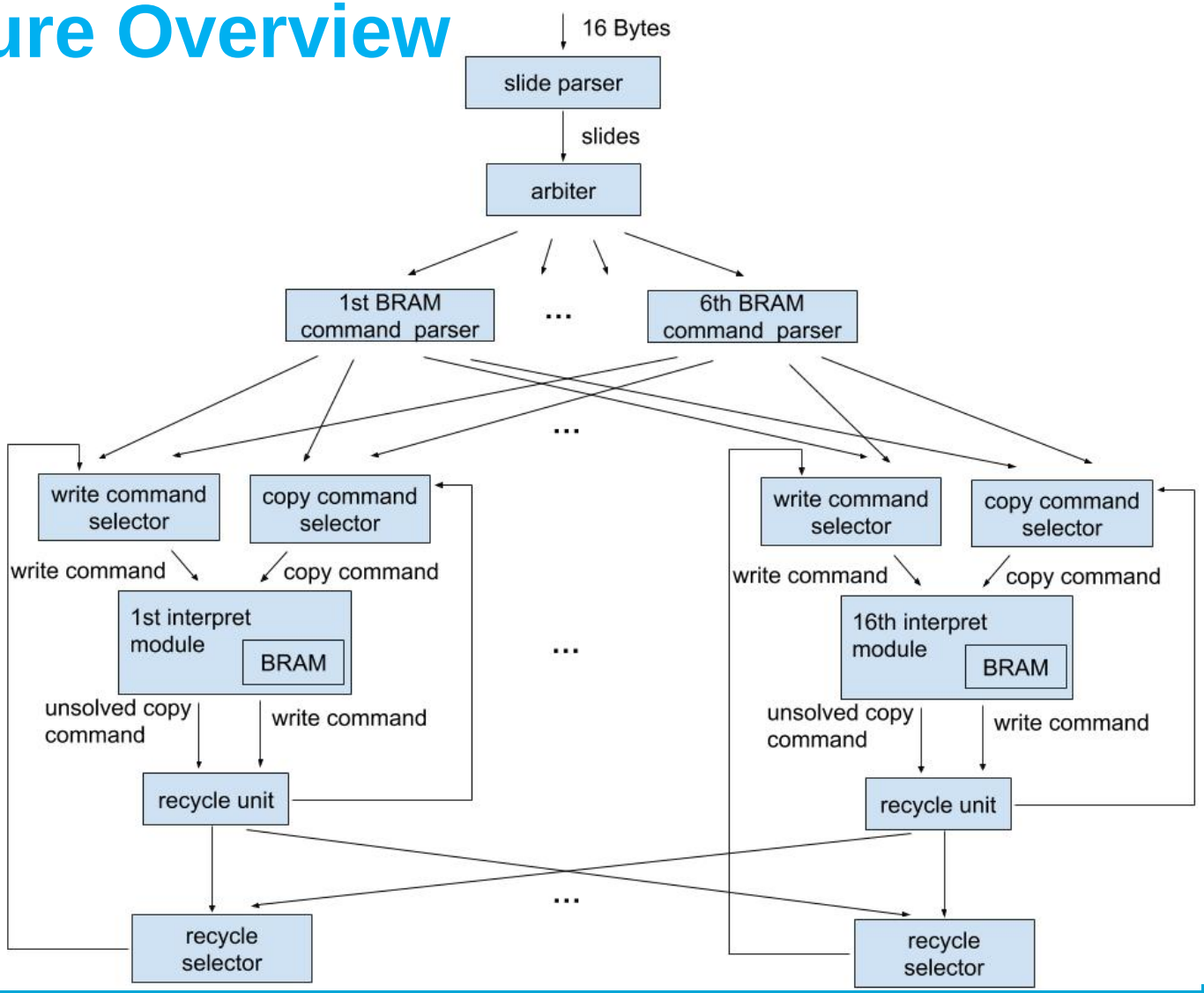
Token size: 4B
2 tokens/cycle
250MHz
--> 13 instances

token size * token/cycle * frequency * instances
= OpenCAPI BW

IDEA – from token-level to BRAM-level



Architecture Overview

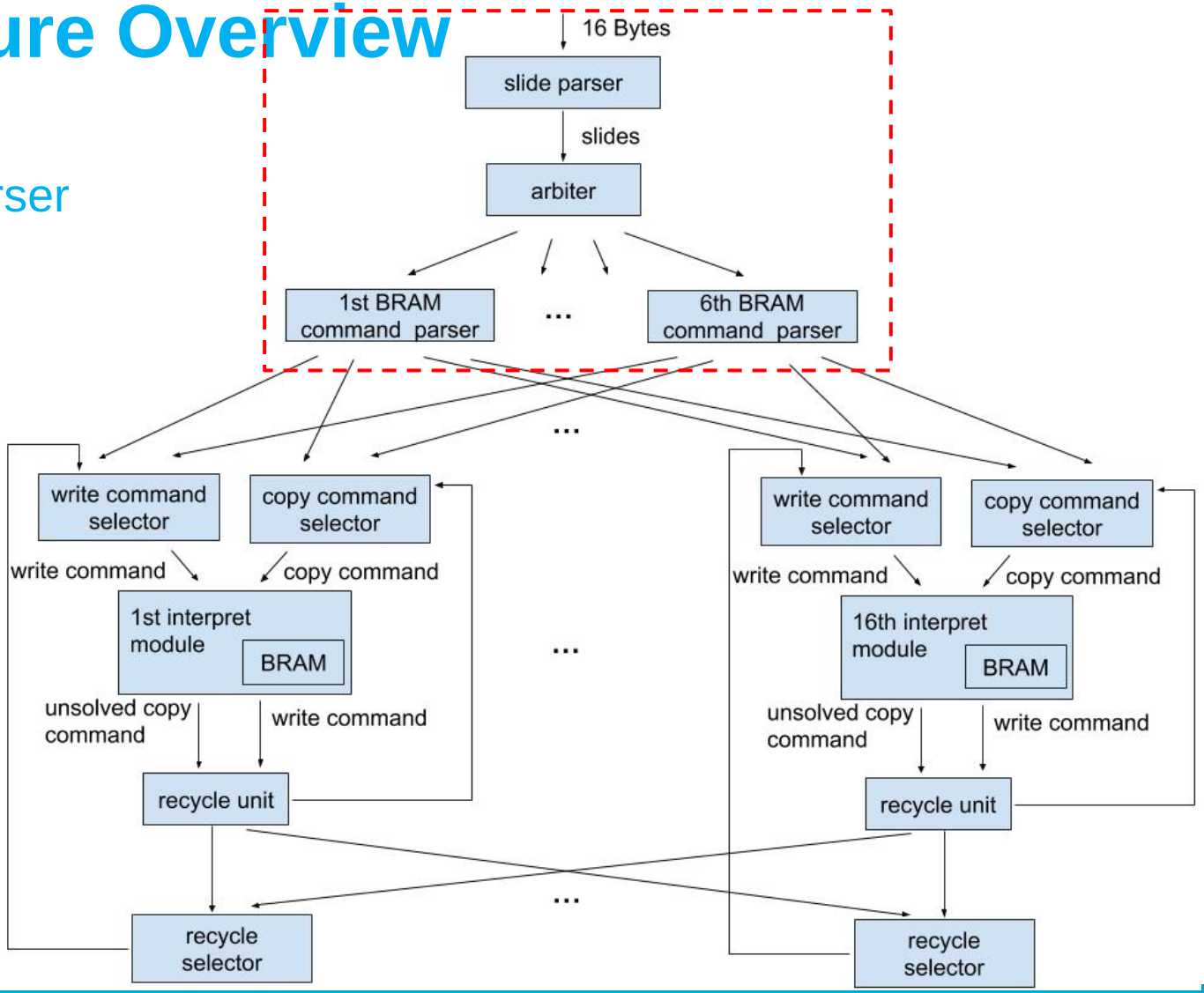


Quantum & Computer Engineering

TU Delft

Architecture Overview

- Two-level Parser

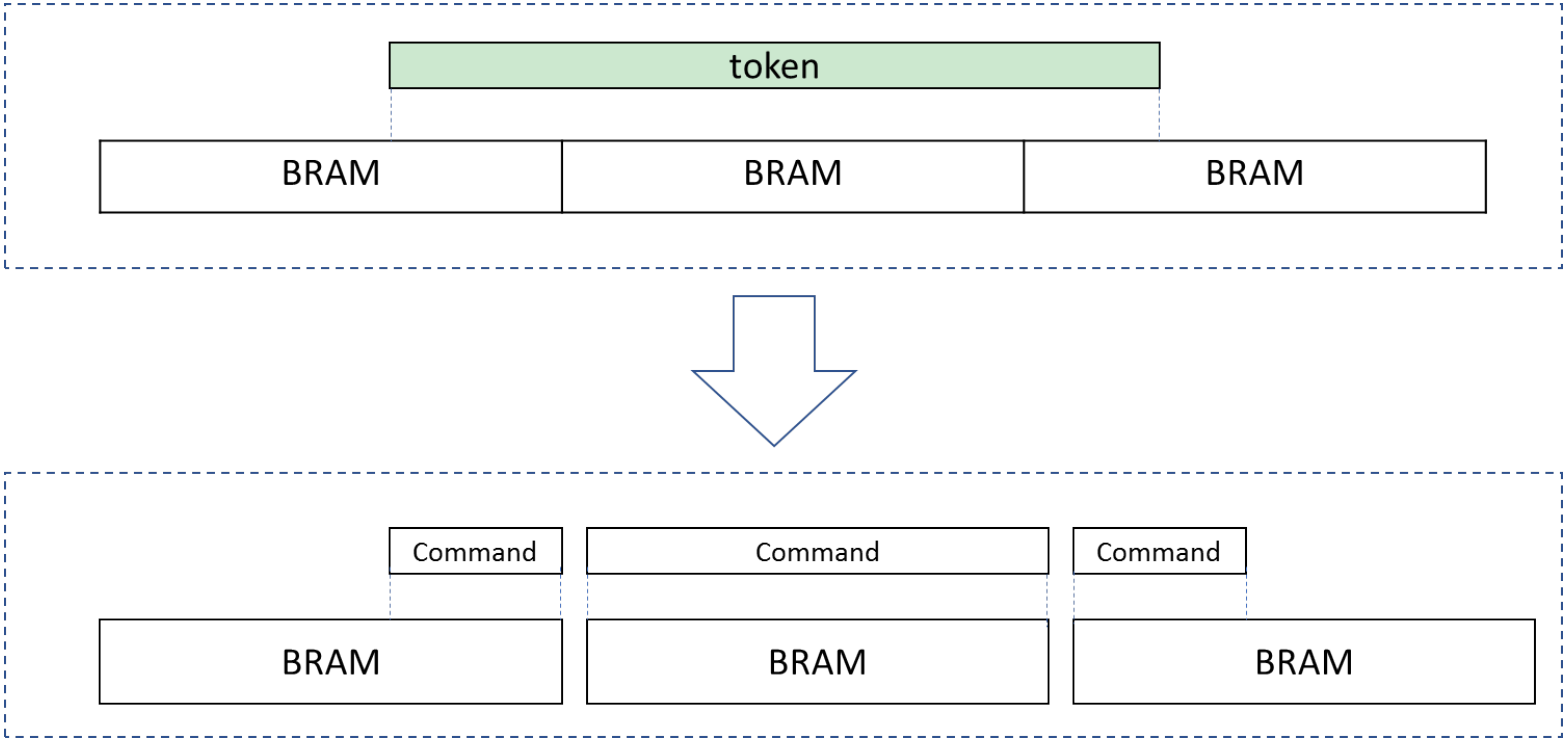


Quantum & Computer Engineering

TU Delft

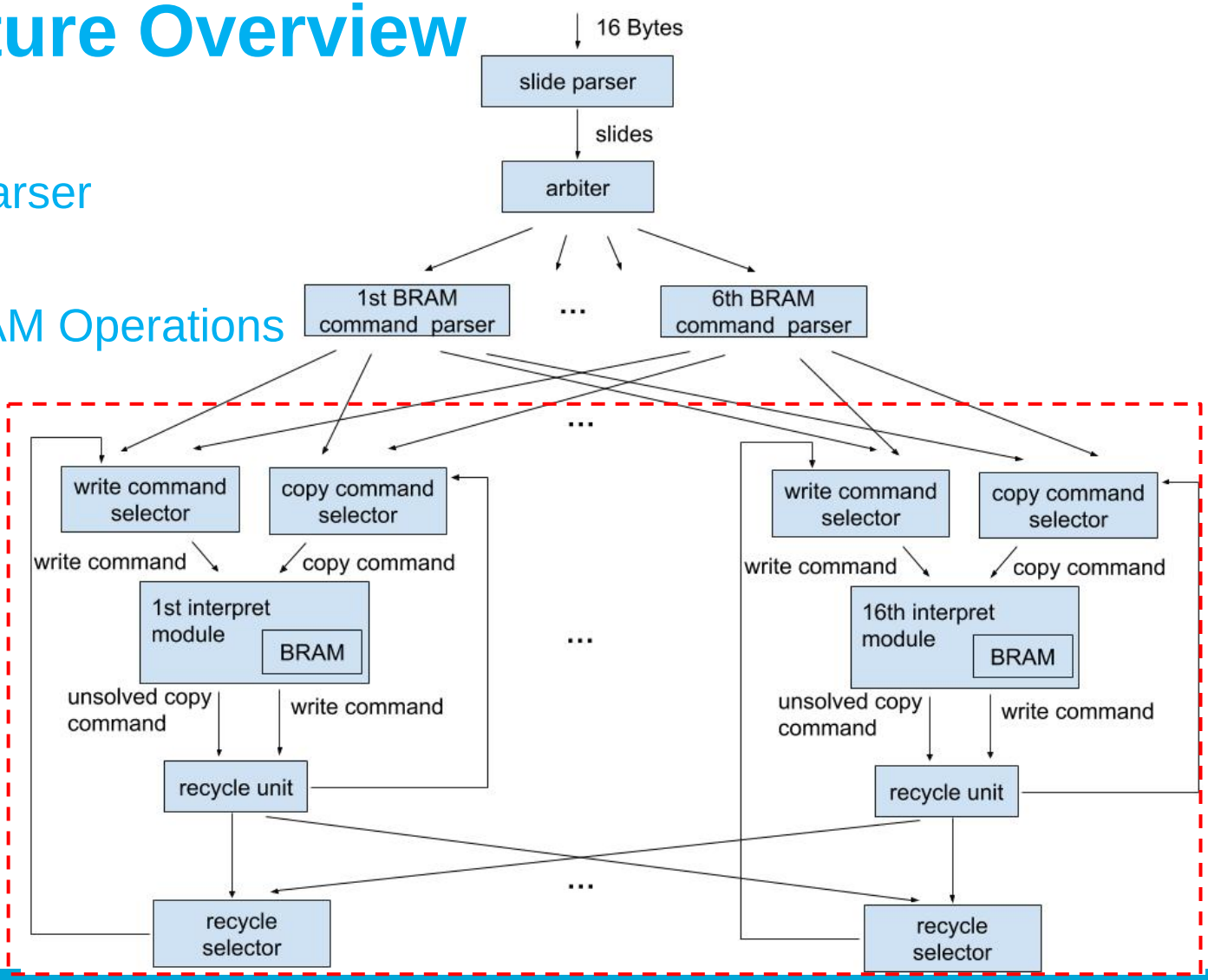
Architecture Overview

- Two-level Parser



Architecture Overview

- Two-level Parser
- Parallel BRAM Operations

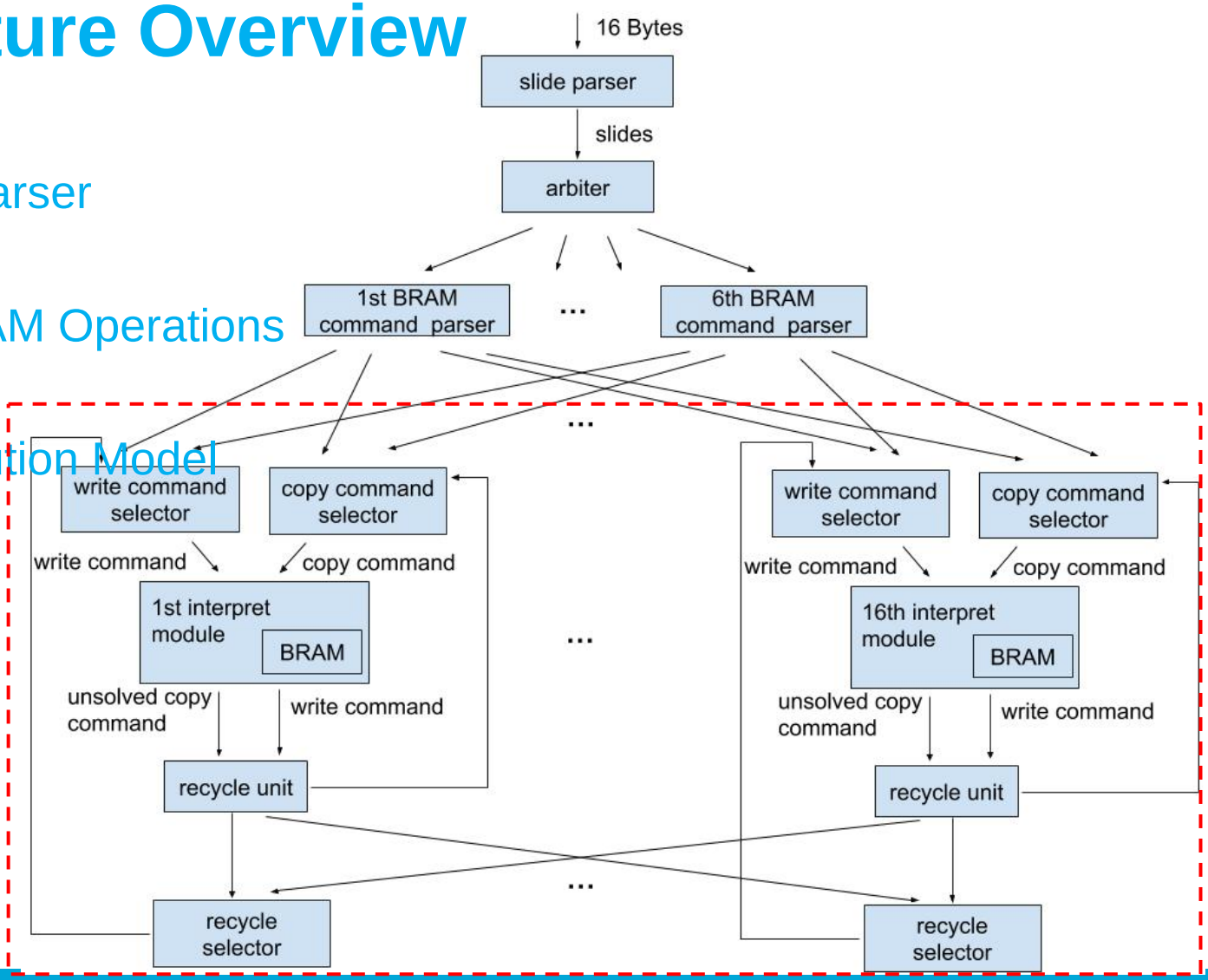


Quantum & Computer Engineering

TU Delft

Architecture Overview

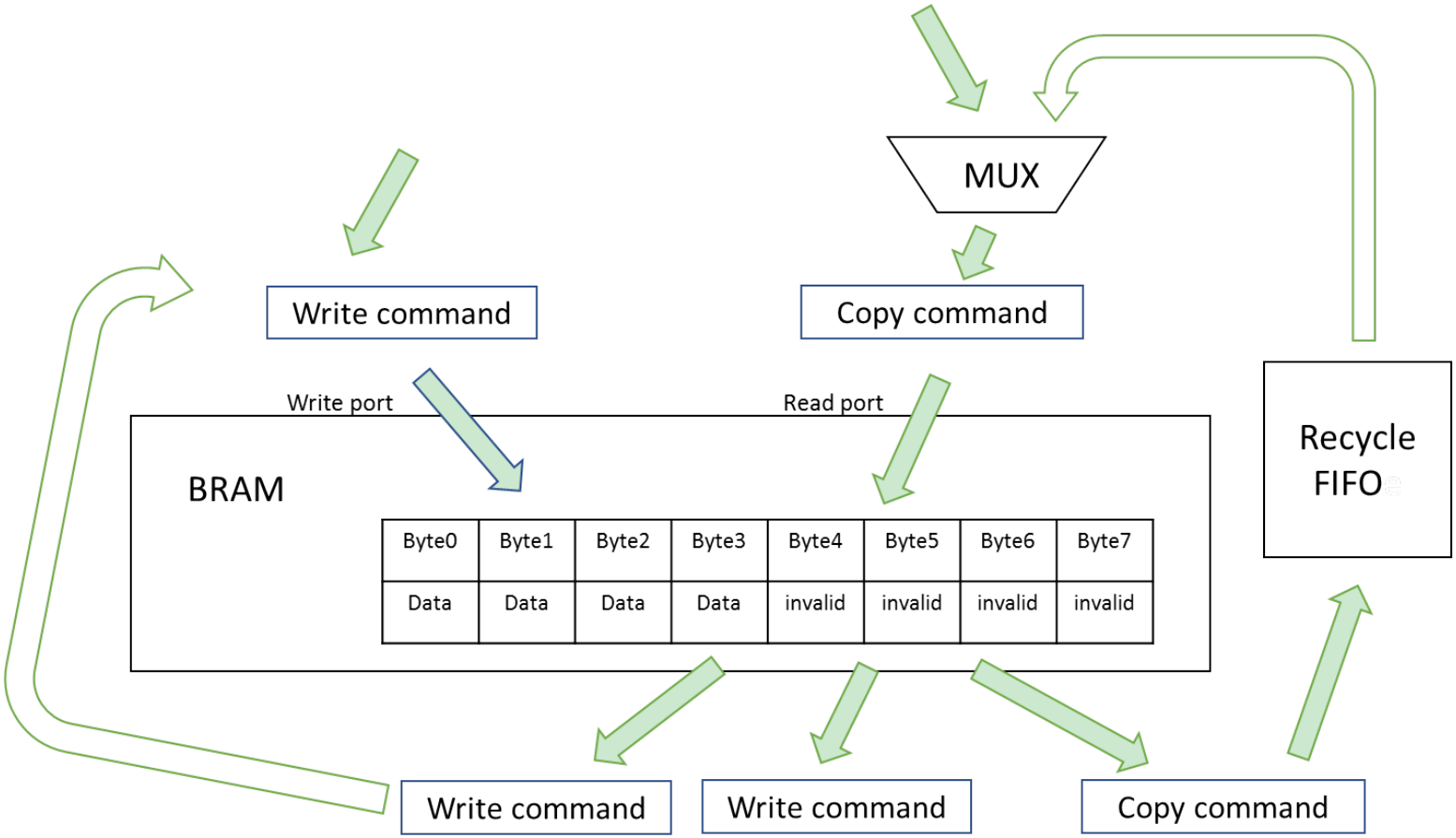
- Two-level Parser
- Parallel BRAM Operations
- Relax Execution Model



Quantum & Computer Engineering

TU Delft

Parallel BRAM Operations & Relax Execution Model



Results

- Implementation Result (on XCKU15P)

Flip-Flop	LUTs	BRAMs	Frequency	Power
32K(3.32%)	46K(8.95%)	29(2.95%)	250MHz	2.9W

- Benchmark
  - Alice's Adventures in Wonderland : 85KB (149KB) for compressed (uncompressed) file.
- Throughput

	FPGA(single decompressor)	CPU(Core i7 with one thread)
Input	3GB/s	300MB/s
Output	5GB/s	500MB/s

Thank You

Publish Paper:

J. Fang, J. Chen, P. Hofstee, Z. Al-Ars, J. Hidders,
A High-Bandwidth Snappy Decompressor in Reconfigurable Logic (CODES+ISSS 2018)

https://github.com/ChenJianyunp/FPGA-Snappy-Decompressor.git

Contact Me:
j.fang-1@tudelft.nl (Jian Fang)

Quantum & Computer Engineering

TU Delft