# Building Data Processing Systems with FPGAs

Spring Semester 2019

Zsolt Istvan (zsolt.istvan@imdea.org)

## Seminar Structure

- This is a highly interactive seminar (similar to a reading group)
  - Presentation: 40%
  - Discussions: 20%
  - Presence: 40%

- We meet 5 times
  - Feb 13 – Introduction
  - Feb 27, March 6, 13, 20 – Presentations

- Each meeting 2h (2 presentation and discussion)
  - Presenters: Read one paper and prepare slides to explain it (20min)

    → Send slides via email 24h before the presentation
  - Others: Read the two papers and prepare for discussion
    - Bullet points of ideas/comments/questions

## Presentation contents

- Presenter: Summarize the paper in an objective way*
  - Explain its context, motivation, what is proposed
  - Explain briefly the implementation techniques
  - Explain how they evaluated their idea and whether it supports their claims
  - + Finish with two opinion slides (see below)
- Presenter & Others: Provide your own opinion about the paper
  - 1) How useful is the proposed thing? Are they solving a real problem? Is the FPGA being used for a genuine reason or could it be done in SW?
  - 2) How convincing is the evaluation? Do they provide support for all claims? Do they compare against related work and state of the art?

Take a look at guides like this

*Feel free to use images, graphs, illustrations from the paper.

## Choice of topics

- List of papers, more than what we can cover – choose subset that looks most interesting for us
  - Link to spreadsheet

- Quick poll familiarity with: Databases? Machine learning? Text processing? Distributed systems?

# FPGAs in Databases

- Today we look at the reason why FPGAs are becoming very relevant in the datacenter

- Quick recap on FPGAs
- Their “traditional” uses in databases
- Three trends that change the status quo
- Challenges ahead

- A copy of the manuscript is here

**A Glass Half Full: Using Programmable Hardware Accelerators in Database Analytics**

Zsolt István
IMDEA Software Institute, Madrid, Spain
{first.lastname@imdea.org}

**Abstract**

*Even though there is a rich body of proposals to accelerate databases using specialized hardware, often the opinion of the community is pessimistic: the performance and energy efficiency benefits of specialization are seen to be outweighed by the limitations of the proposed solutions and the additional complexity of including specialized hardware, such as field programmable gate arrays (FPGAs), in servers. Recently, however, as an effect of stagnating CPU performance, server architectures started to incorporate various programmable hardware components, ranging from smart network interface cards, through SSDs with offloading capabilities, to near-CPU accelerators. This availability of heterogeneous hardware brings opportunities to databases and we make the case that there is cause for optimism. In the light of a shifting hardware landscape and emerging analytics workloads, it is time to revisit our stance on hardware acceleration.*

*In this paper we highlight several challenges that have traditionally hindered the deployment of hardware acceleration in databases and explain how they have been alleviated or removed altogether by recent research results and the changing hardware landscape. We also highlight that, now that these challenges have been addressed, a new set of questions emerge around deep integration of heterogeneous programmable hardware in tomorrow's databases, for which answers can likely be found only in collaboration with researchers from other fields.*

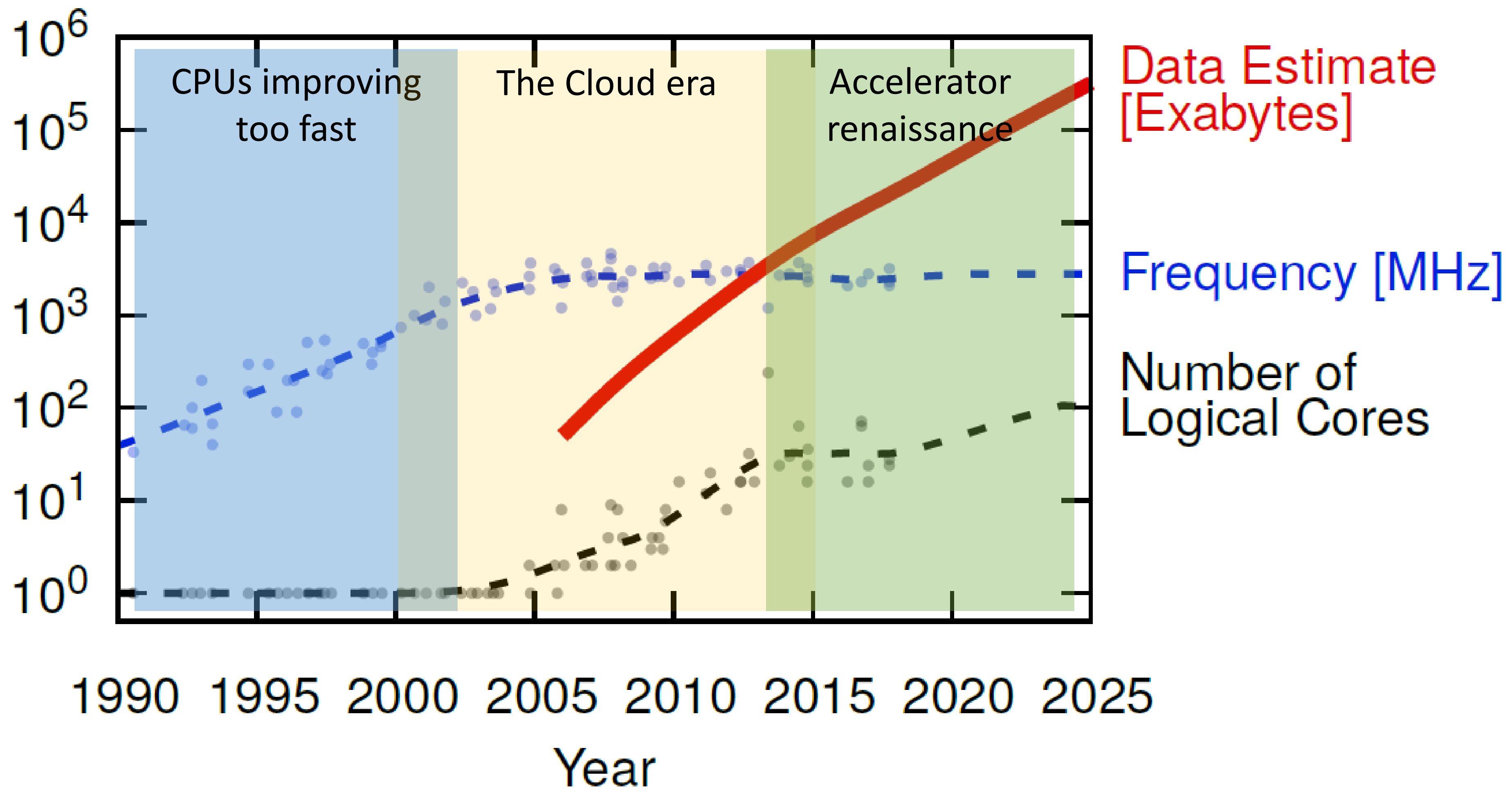
**1 Introduction**

There is a rich history of projects aiming to specialize parts of, or entire, computers to databases. Notable examples include the Database Machine from the seventies [1], Gamma [2], the Netezza data appliance [3], the Q100 DB processor [4], and Oracle Rapid [5] most recently. These works demonstrate dramatically increased energy efficiency and better performance thanks to a hardware/software co-design approach. However, CPUs, until very recently, enjoyed a performance scaling in line with Moore's law and the time and effort of designing and delivering specialized hardware was not economical. This changed with the stagnation in CPU performance [6] in the last decade and the simultaneous increase in networking speeds that has created a clear need for hardware acceleration.

Initially, the move to the cloud worked against hardware acceleration for databases due to the cloud's reliance on commodity hardware and the need to cater to many different users and applications. In the meantime,

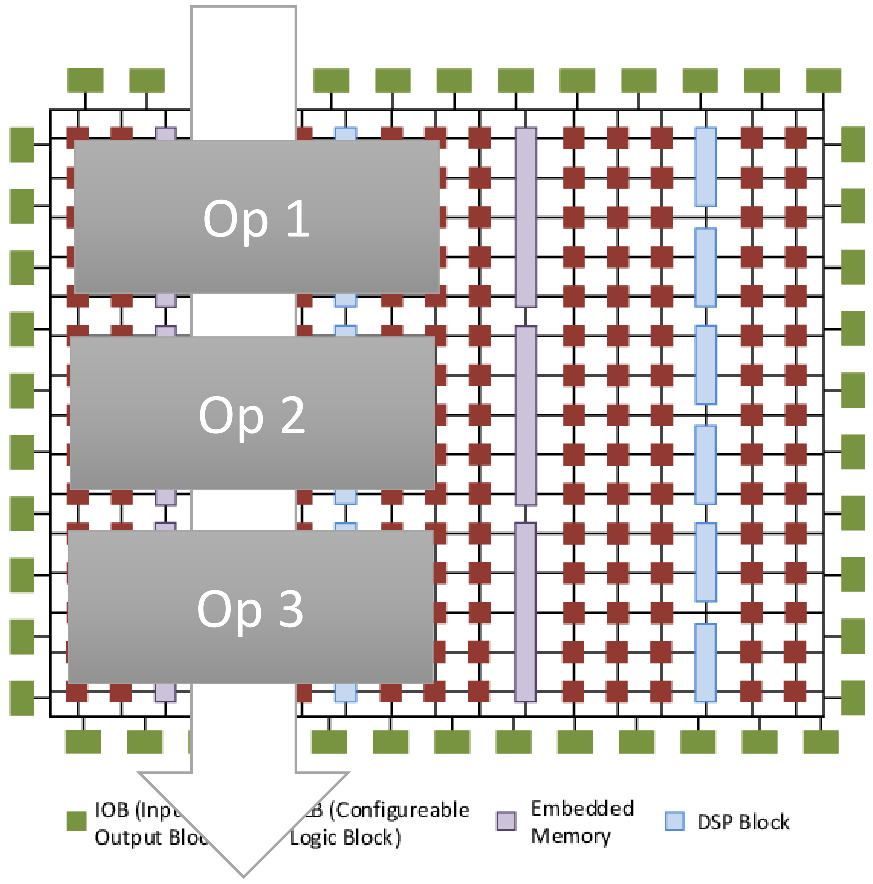
**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

## Data/Compute Gap



Based on a plot layout by K. Rupp. Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten; 2010-2015 by K. Rupp. Data growth estimate by C. Maxfield.

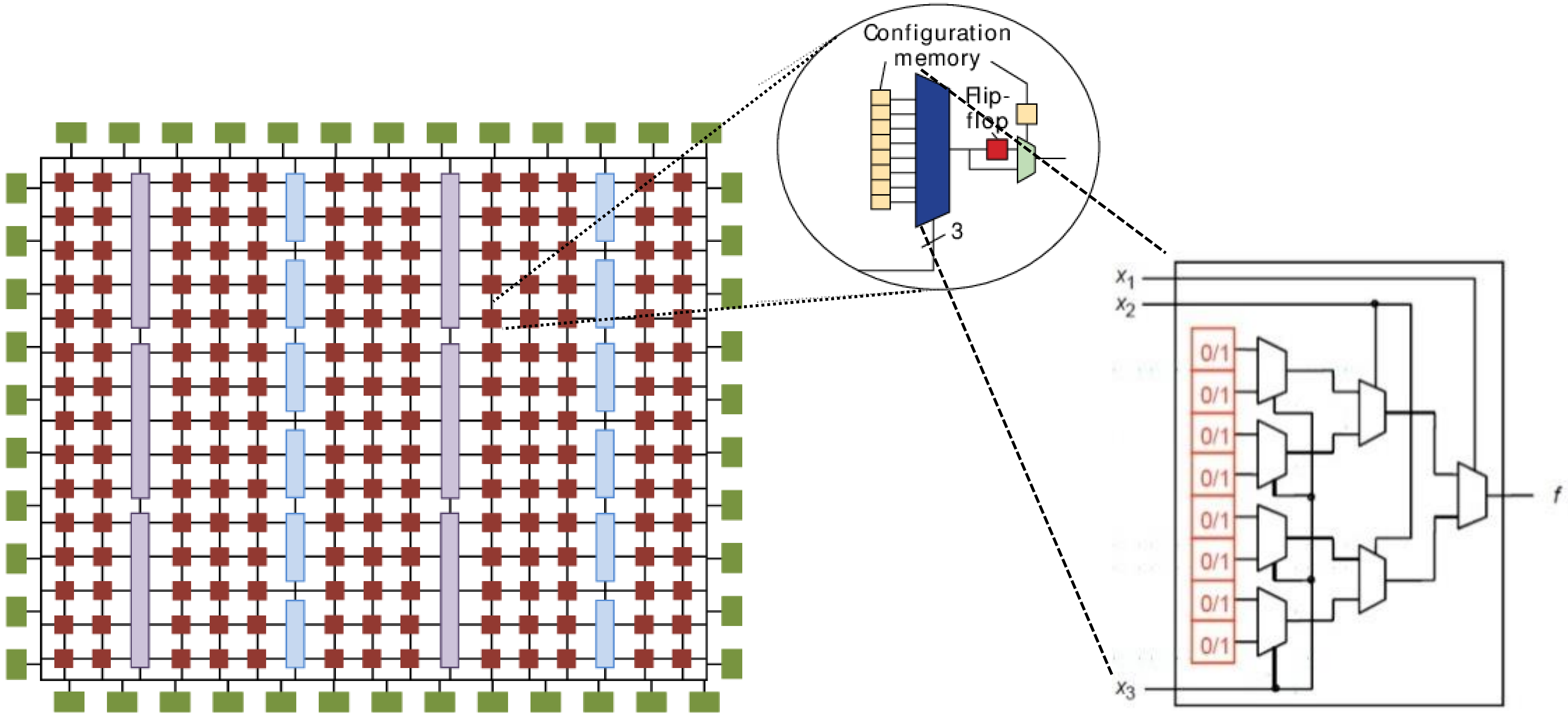
# Re-programmable Specialized Hardware



**F**ield **P**rogrammable **G**ate **A**rray (FPGA)

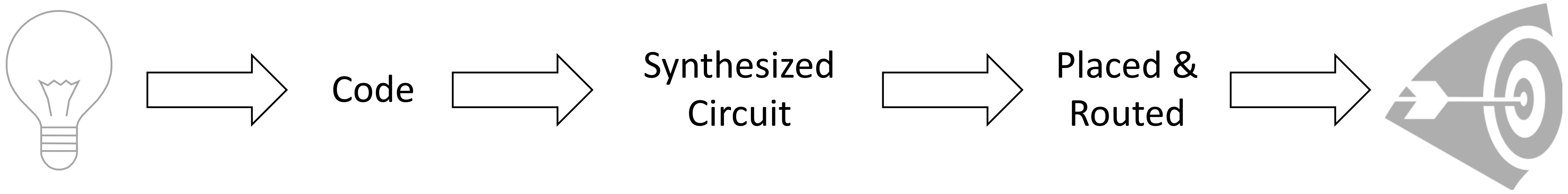
- Free choice of architecture
- Fine-grained pipelining, communication, distributed memory
- Tradeoff: all “code” occupies chip space

## What is inside a Logic Block?



🟩 IOB (Input/ Output Block) 🟥 CLB (Configureable Logic Block) ▨ Embedded Memory ▨ DSP Block

# Programming FPGAs

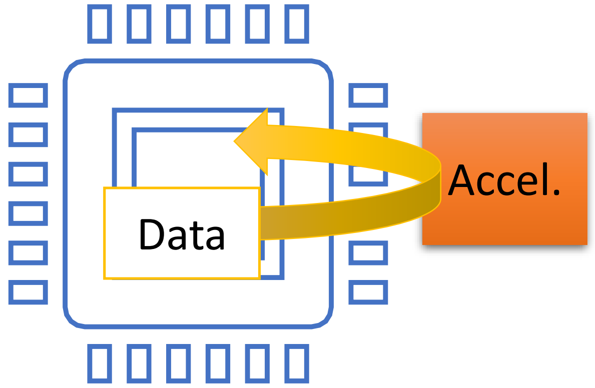


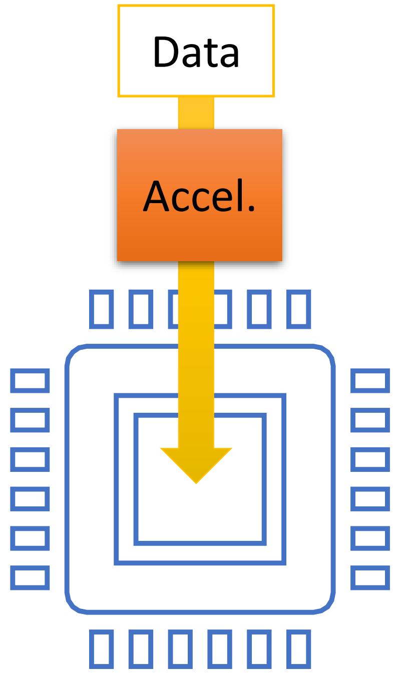
- Coding: Hardware definition languages, high level languages
- Synthesis: Produce a logic-gate level representation (any FPGA)
    - Seconds – minutes
- Place & route: Circuit that gets mapped onto specific FPGA
    - Minutes – hours

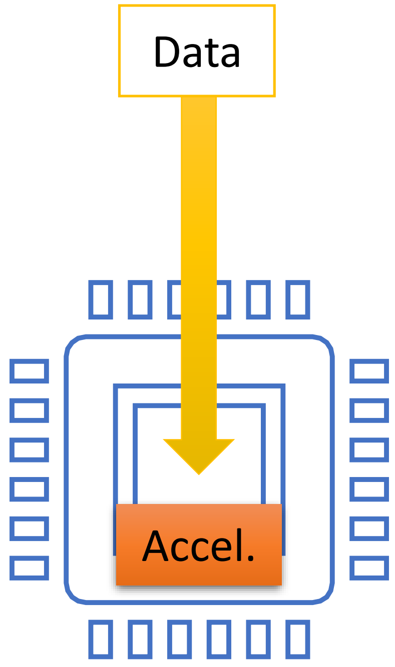
## FPGA Refresher

- For more context watch these videos
  - An overview of FPGA history
  - Microsoft deploying FPGAs in Azure
  - Regular expressions of FPGAs (skip first 7 minutes)

# Where to position specialized hardware?



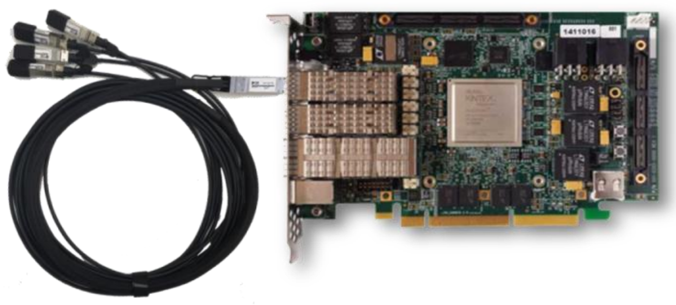


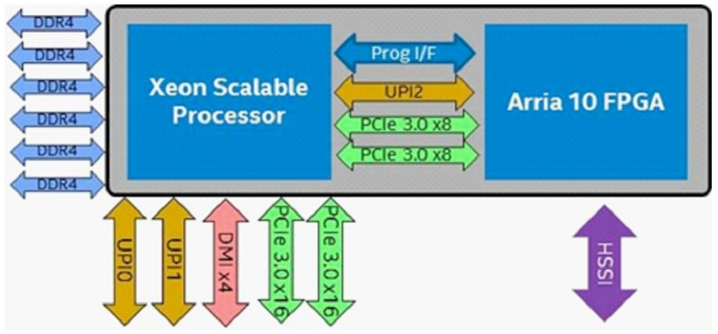


**1) On the side**

**2) In data-path**

**3) Co-processor**





# On the side acceleration

- Similar to GPUs
  - Accelerator accessed over interconnect (PCIe)
  - Latency, data copy and bandwidth overhead (the latter has improved)
  - Makes sense for compute-intensive workloads ($CPU_{BW}<FPGA_{BW}$)

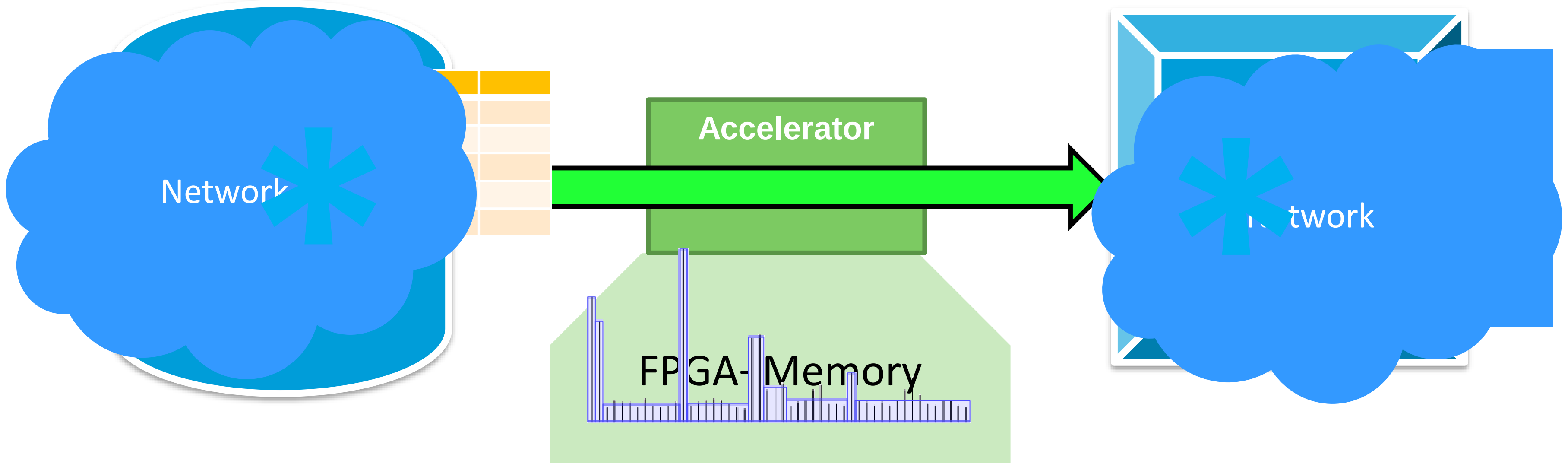
- Why is data copy/transformation needed?
  - Main memory data structures might have to flattened, extra data removed
- Side-effect of latency and transform overhead?
  - Pushing to coarse grained offloading
  - Might become “too much” for FPGA
  - Question: what does the CPU do while waiting for the FPGA?

## On the side acceleration (II)

- Promising areas today
    - Machine learning, genome sequencing, etc.
- Integration effort with software
    - Moderate

## Histograms for databases

- Statistical accelerator in the data path
- Hardware implementation that delivers the same type of histograms as databases without sampling

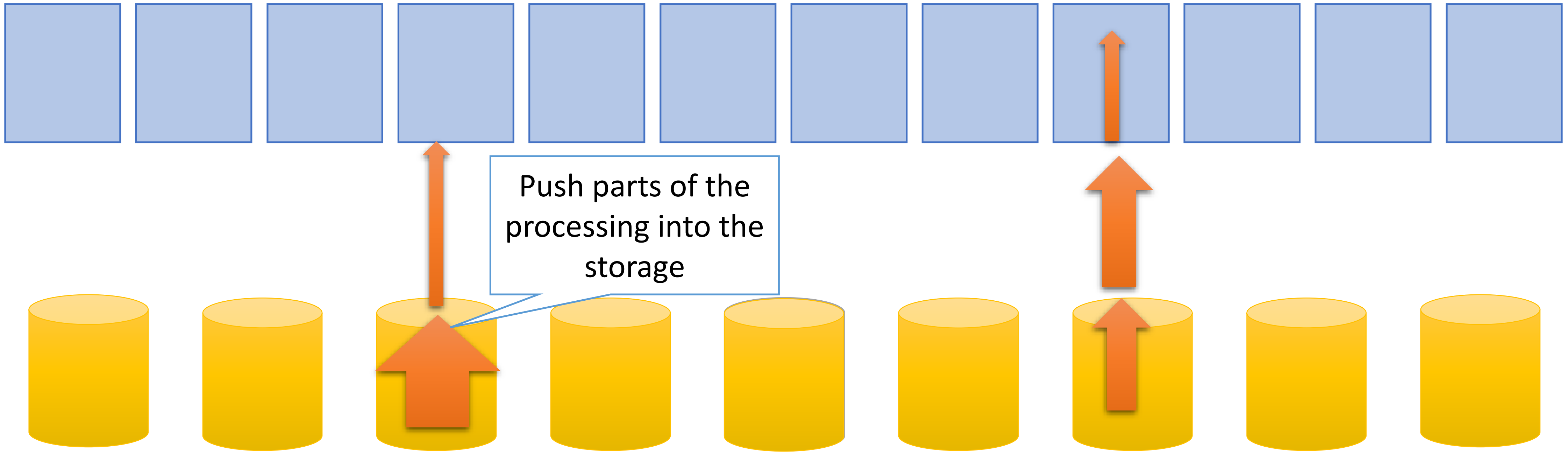


# In data path acceleration

- Processing between data source and “main processor”
  - Goal: Reduce data amount or provide added value **without** slowing down data retrieval
- Is latency not an issue?
  - Data is moved anyway from the source (think NVMe flash, distributed storage)
- FPGA circuits designed to provide fixed bandwidth operation (line-rate)
  - Research question: how to re-imagine compute-bound algorithms to be memory bound?

# In data path acceleration (II)

1) Bandwidth larger inside the storage than on the network
2) Not all data required for answering all queries
3) Value added services can be performed “for free”

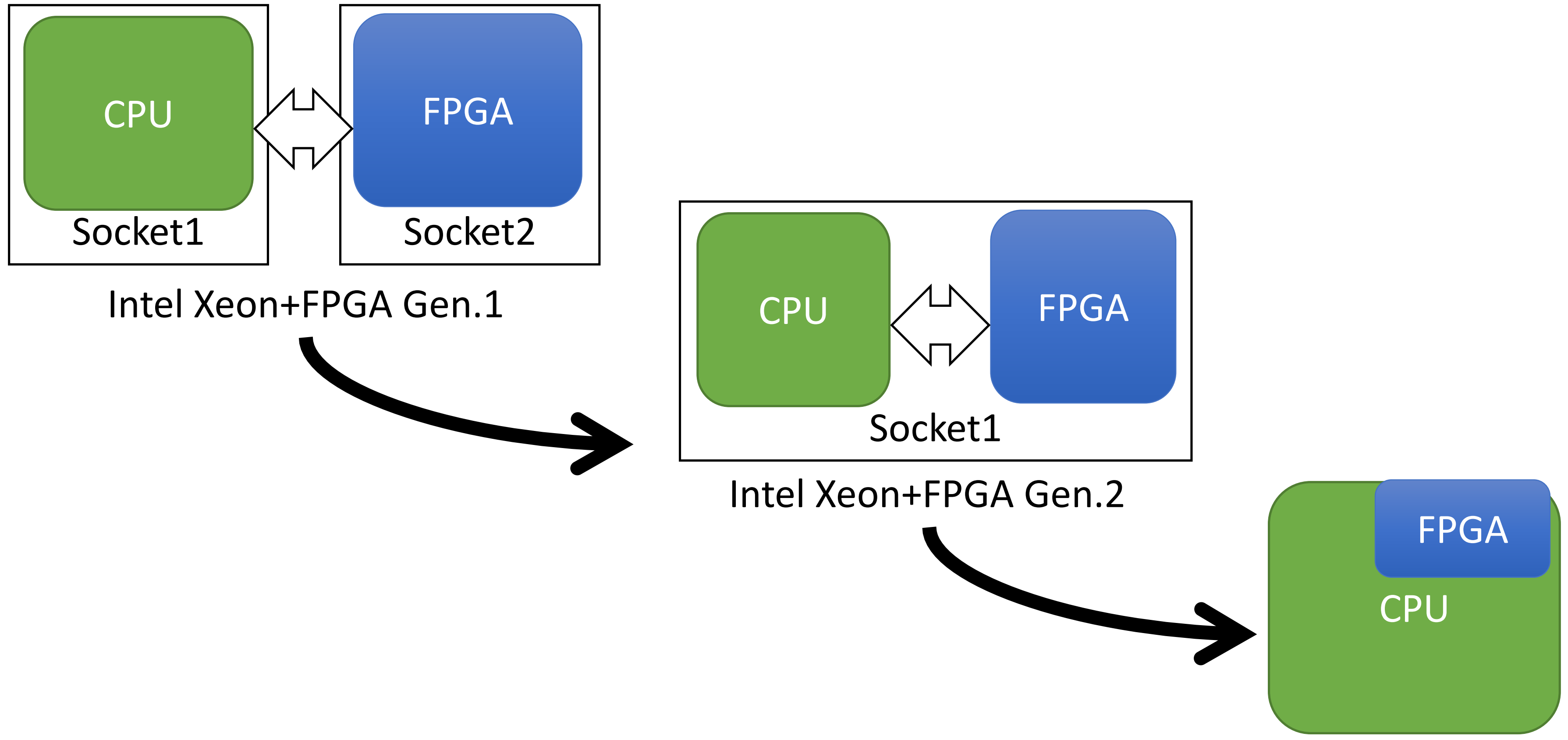


# In data path acceleration (III)

- Promising areas today
  - Compression, decompression, data transformations, filtering, …
  - Often in front of (distributed) storage

- Integration with software
  - Fairly decoupled but sometimes has to be able to parse app-specific formats/packets

# Intel Xeon+FPGA



# Co-processor

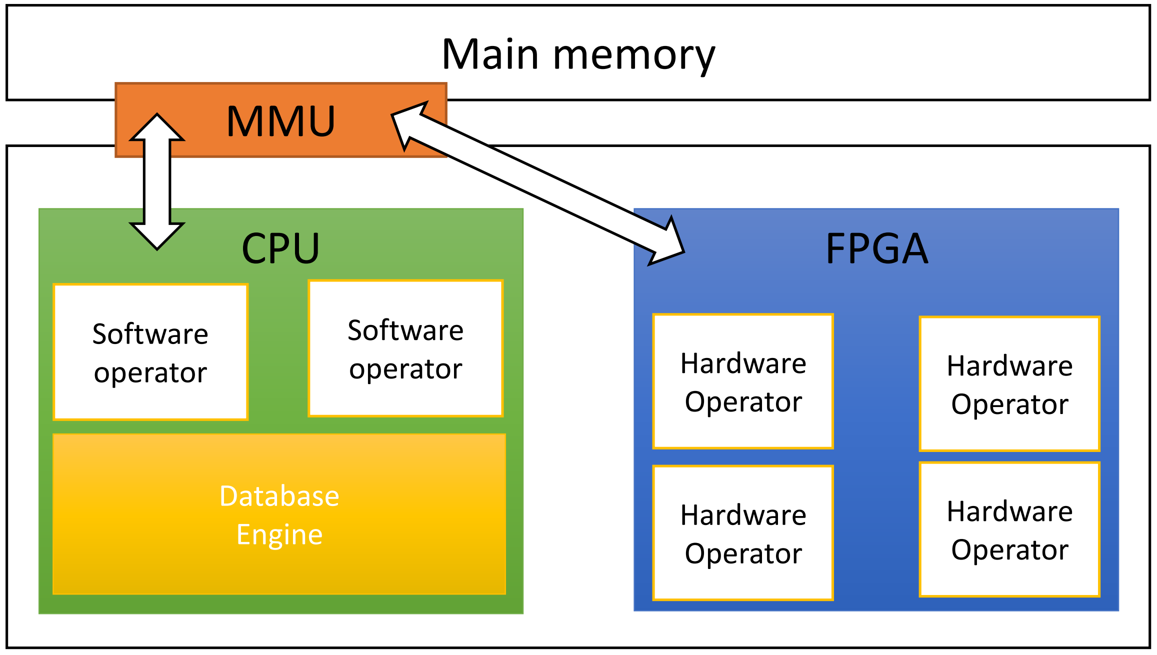
- FPGA can access same memory as CPU
    - Cache coherency
    - Supports virtual memory
    - Acts like an other CPU/core
- Latency overhead?
    - Negligible, in the nanoseconds
- Copy/transformation overhead?
    - Data can be accessed directly
    - But smaller bandwidth than CPU (e.g., 12GB/s vs. 50GB/s)

# Co-processor (II)

- Promising areas?
  - Compute-intensive operations
  - Parts of operators can be offloaded (e.g. hashing of a hash join in databases)

- Integration
  - Tight, more challenging

- In some ways similar to "on-the-side" but negligible overhead



DoppioDB

# TODO for next days/weeks

- Please enter your preferences for the papers before Sunday 17th of February
  - We will post schedules online (and email you)

- Start preparing your talks, even if you are not among the firsts
  - Email me the slides to your presentation 24h in advance

- Read the papers for the upcoming week and take notes to help you with discussion
  - When thinking of the papers, recall the types of acceleration and their better/worse applications