Accelerating the merge phase of sort-merge join

FPL 2019 – The 29th International Conference on Field-Programmable Logic and Applications

Philippos Papaphilippou, Holger Pirk, Wayne Luk

Dept. of Computing, Imperial College London, UK
`{pp616, pirk, w.luk}@imperial.ac.uk`

Source code: philippos.info/mergejoin

The task: equi-join

A-Key	Value
A1	2
A2	2
A3	3
A4	3
A5	3
A6	11

$\bowtie$

B-Key	Value
B1	2
B2	2
B3	3
B4	5
B5	6

$=$

A-Key	B-Key	Value
A1	B1	2
A1	B2	2
A2	B1	2
A2	B2	2
A3	B3	3
A4	B3	3
A5	B3	3

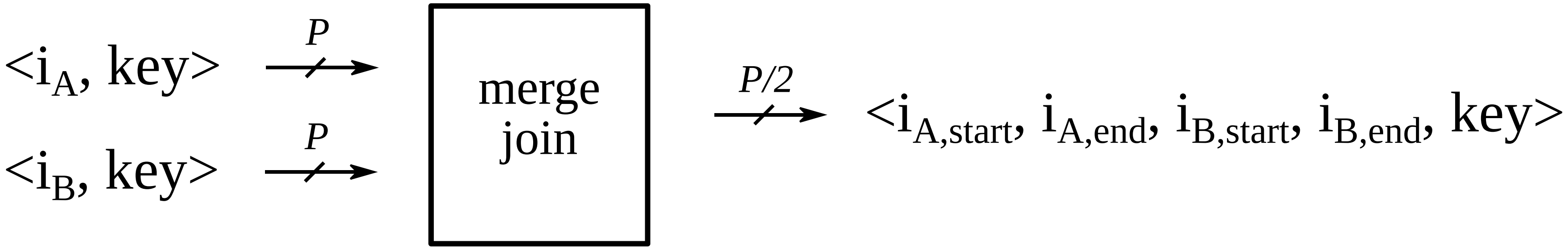
- Equi-join
  - Join two tables based on key equality
  - Cartesian product when there are more than 1 keys in one of the 2 tables
- Popular algorithms
  - Hash-join → *Random access pattern*
  - Sort-merge join → *Streaming access pattern → FPGA friendly*

Challenges in related work

- Input properties
  - Presence of duplicate keys → complicates the hardware and access patterns
  - Long input → limited storage inside the FPGA
  - Wide input → moving big rows is expensive
  - Some designs are inapplicable or slow down
- Data movement
  - Narrow inter-chip (CPU ↔ FPGA) communication
  - Induced latency
- Scalability
  - Future technologies (High-throughput)
  - Big data → arbitrarily long tables

Abstracted solution

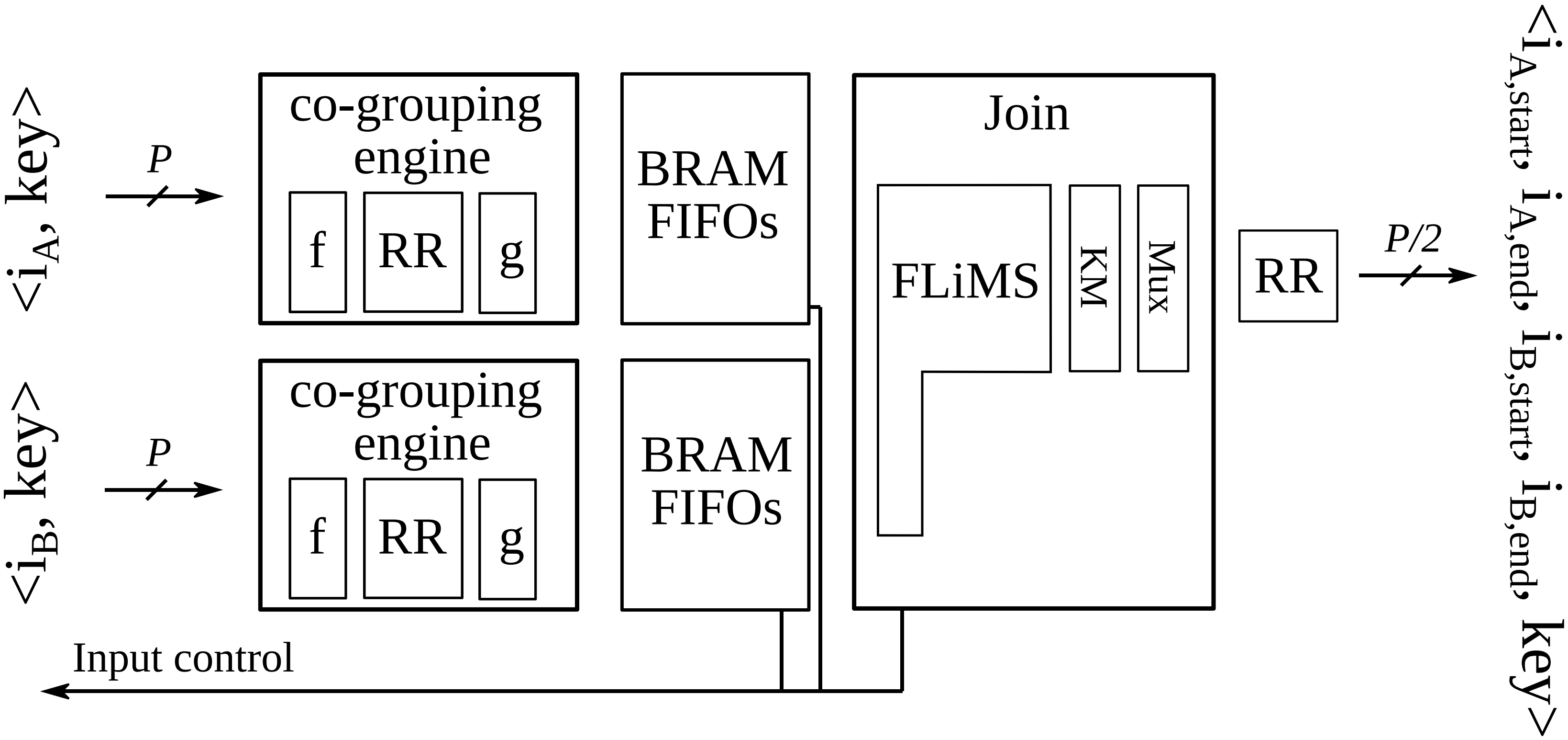
- High-Throughput Stream processor
- Inputs
  - Sorted keys of table A
  - Sorted keys of table B
- Output
  - Index ranges where the key was the same
- Expand on demand (late materialisation)



Proposal

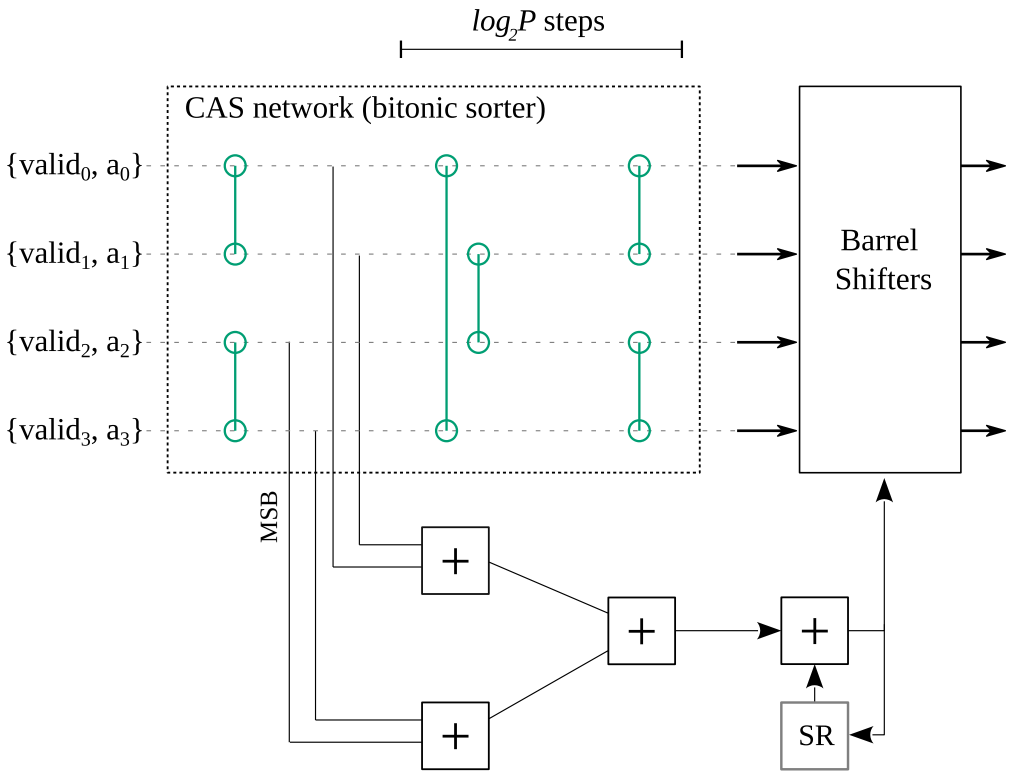
Building blocks

- Round-robin module
- Co-grouping engine
- Modified FLiMS



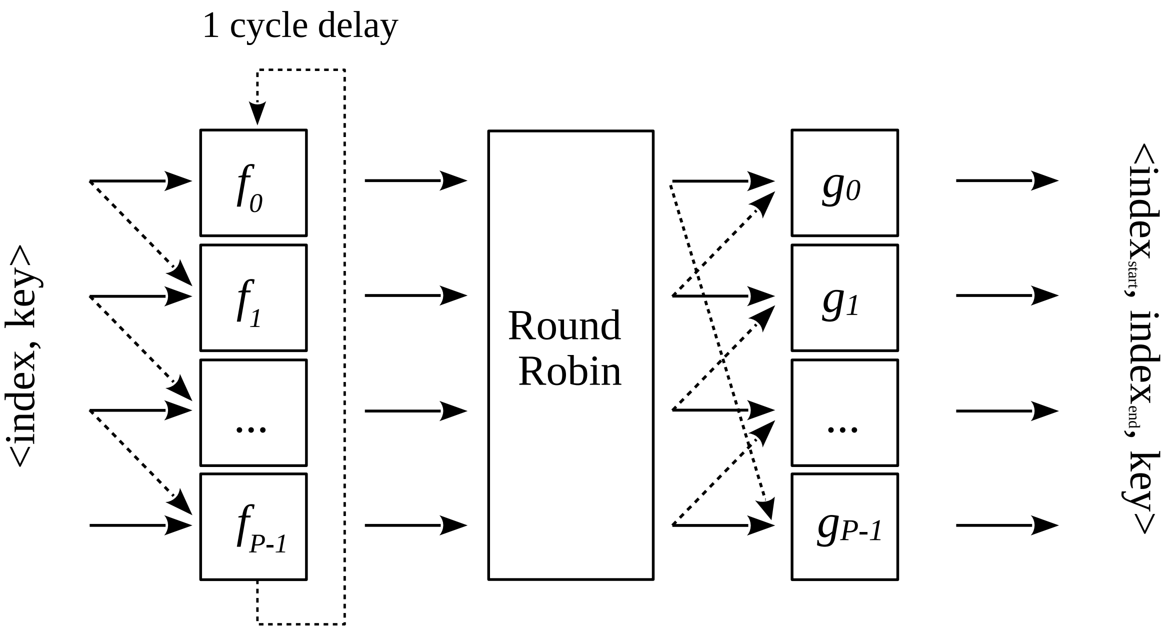
Round-robin module

- Stream processor
- Rearranges sparse input, before writing in multiple banks
- Round-robin effect, but in parallel



Co-grouping engine

- Stream processor
- Provides ranges of indexes, where the key was the same
- Input: Sorted keys
- Output: Unique keys, index ranges



Join module

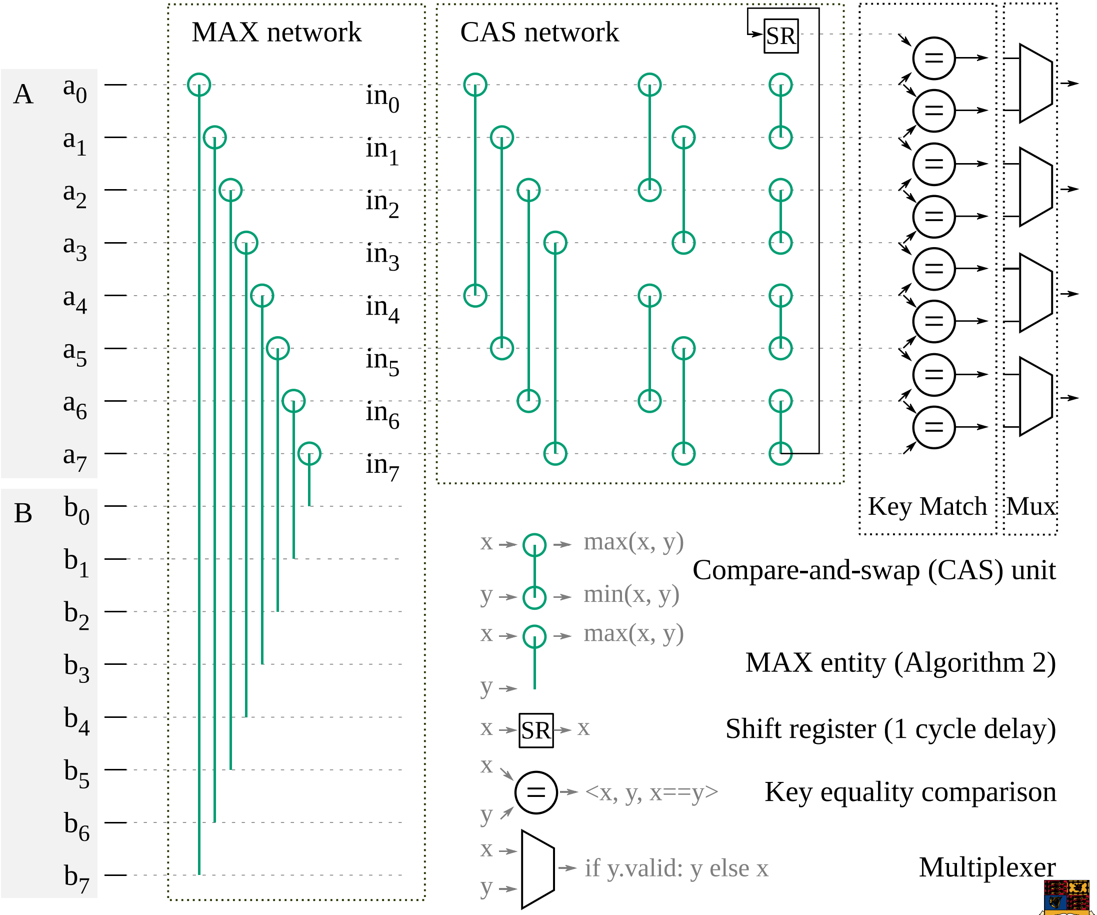
- Task: merge 2 co-grouped streams
- Output: tuples of the form

  <index$_{Astart}$, index$_{Aend}$,

  index$_{Bstart}$, index$_{Bend}$,

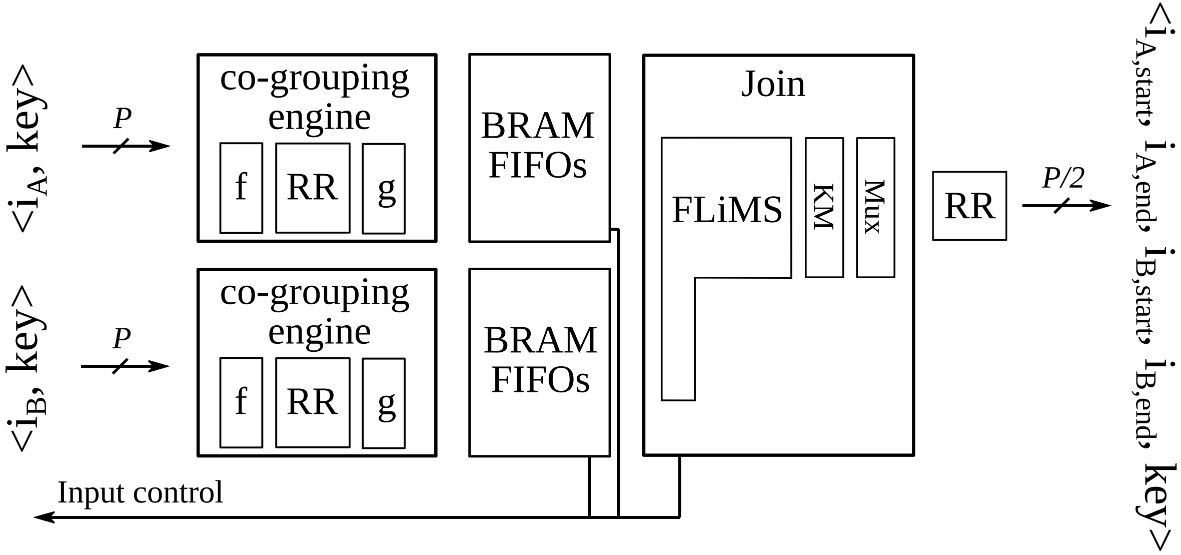
  key>

- Main idea:
  - Sort them together
    - Based on a high-throughput H/W merge sorter (FLiMS [FPT'18])
  - Match same-key groups, by only looking at consecutives



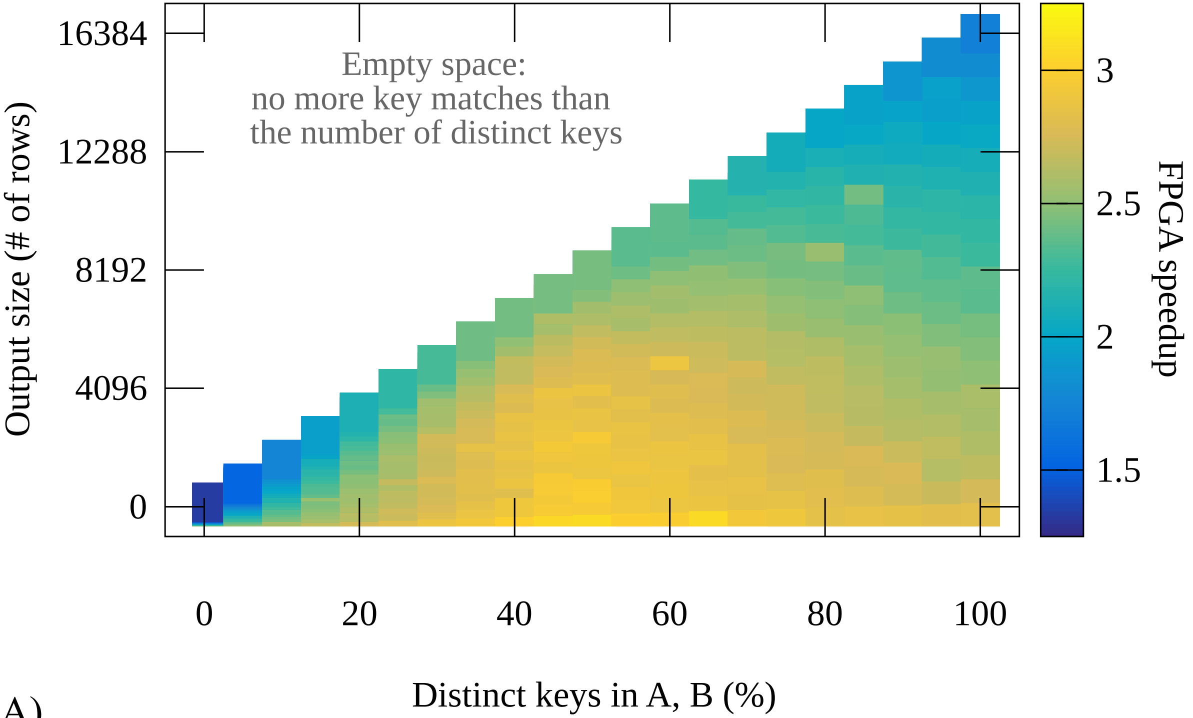
Advantages

- Input agnostic
  - Index-based
  - Big data analytics
- Stream processor
  - FPGA-friendly
- Modular design
  - Novel building blocks
  - Can be combined with other: H/W sorters, filters, ...
- High-throughput design
  - Scalable for future architectures
  - Lower resources than related work



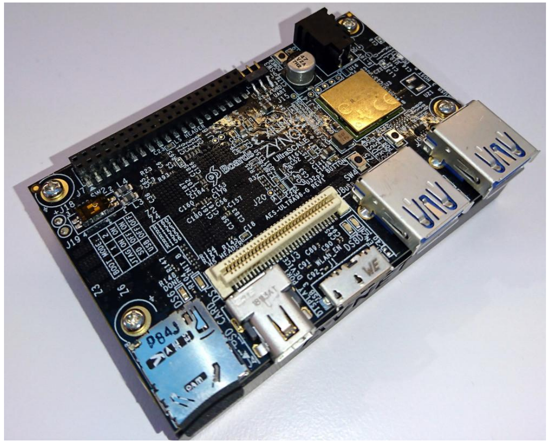
Evaluation on a heterogeneous system

- Platform
  - Zynq UltraScale+ device
  - Operating system: Petalinux
  - Communication: DMA transfers
- Speedup of up to 3.1 times
  - 1-port (H/W) vs 1-thread (S/W)
- Input design space exploration
  - Fraction of distinct keys (%)
  - Fraction of key matches (%) (directly related to the output size)
- Speedup variation factors
  - CPU performance
  - Length of the DMA transfers (CPU→ FPGA)



END

Thank you for your attention!



Source code for Ultra96:

`philippos.info/mergejoin`