Hardware Acceleration of Database Operations

Jared Casper and Kunle Olukotun

Pervasive Parallelism Laboratory

Stanford University

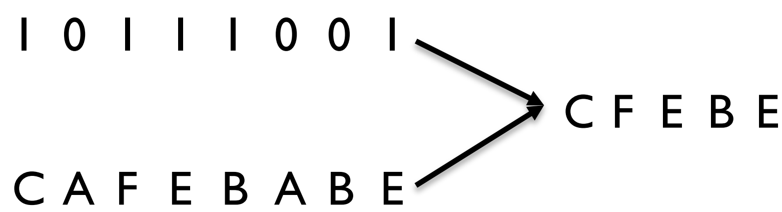
Database machines

- Database machines from late 1970s
  - Put some compute on the disk track/head/unit
  - Processors got faster, I/O performance did not
  - Processor could keep up with disk
    - No performance left on the table
- Today's database machines
  - Made up of general purpose components
  - Massive amounts of memory
  - Very high speed interconnect
  - Tables, even databases, fit entirely within memory

Database Operation Acceleration

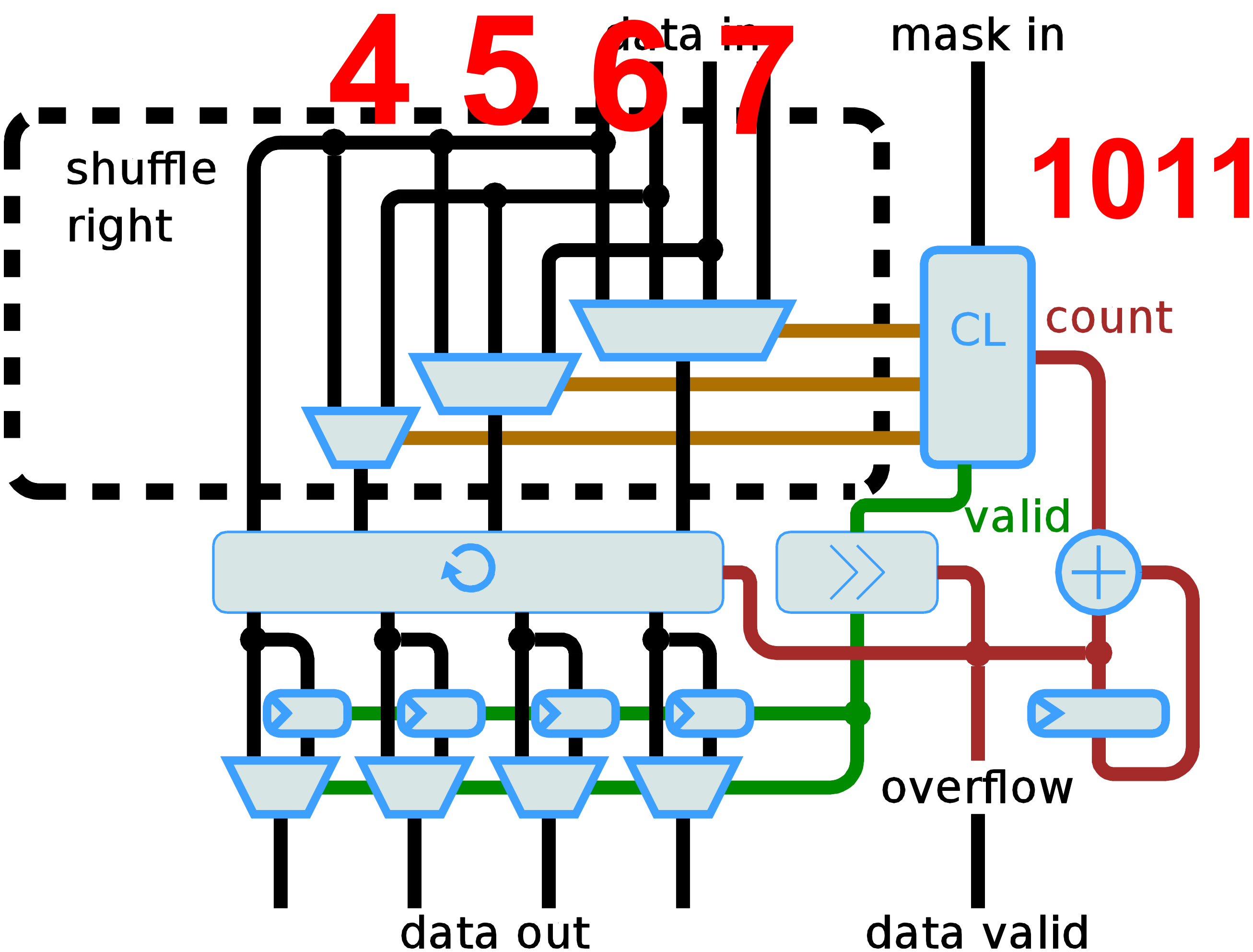
- Processors can not keep up with memory
  - Join performance is at 100s of million tuples per second
  - 64-bit tuples → 2-3 GB/s
  - Chips can get over 100 GB/s
  - Performance is being left on the table
- Follow 10x10 rule, build accelerators
- Three acceleration blocks
  - Selection, merge join, sort
  - Combine these to do a sort merge join
  - Goal is to "keep up with memory"

Select



- Software implementation uses SIMD
  - Read data into SIMD register
  - Use SIMD shuffle operation to move selected data to one end of the register
    - Mask used as index into table for shuffle values
  - Unaligned write to append to output
  - Limited by SIMD width, number of SIMD registers

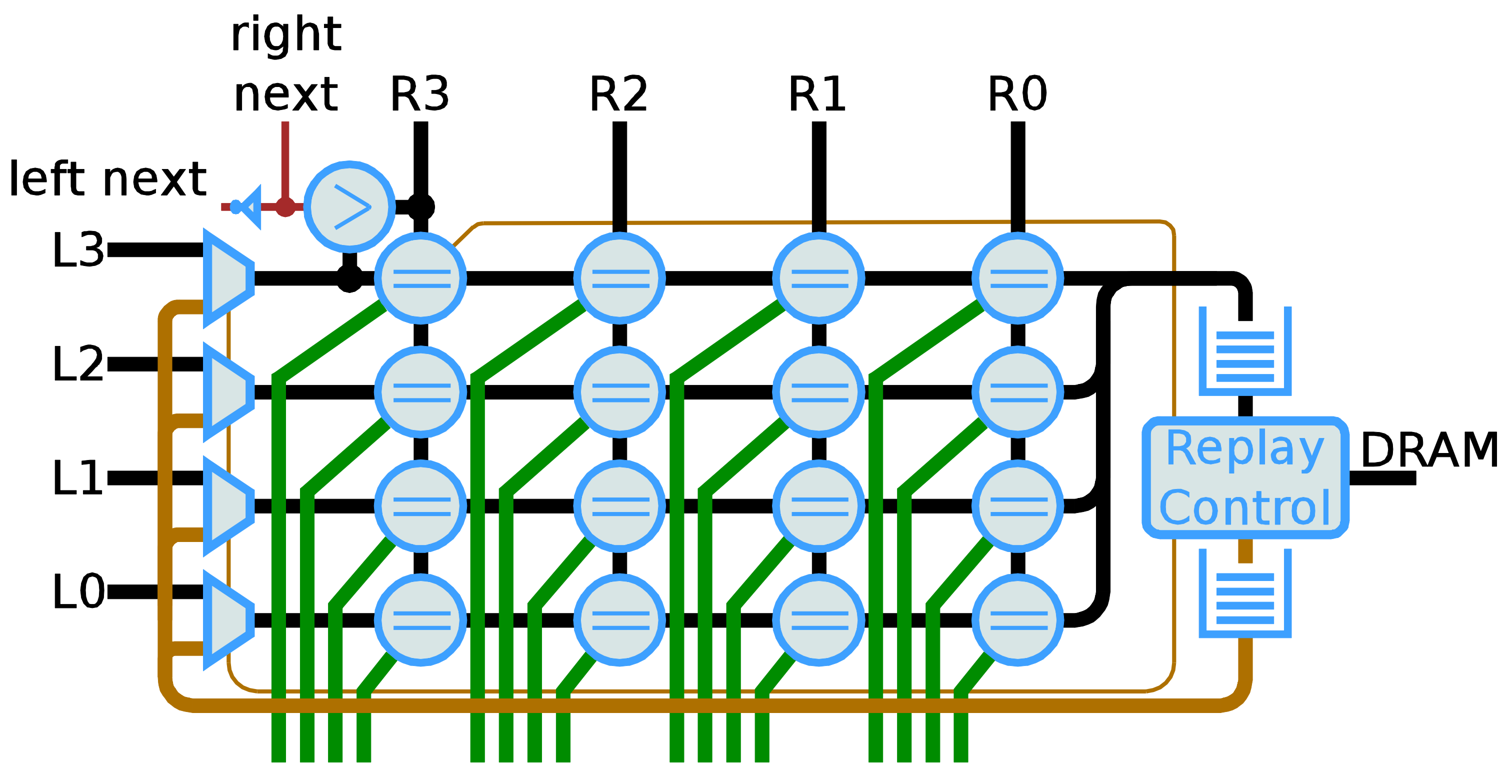
Select


4 5 6 7
data in
mask in
1011
shuffle
right
CL
count
valid
overflow
data out
data valid

Merge Join

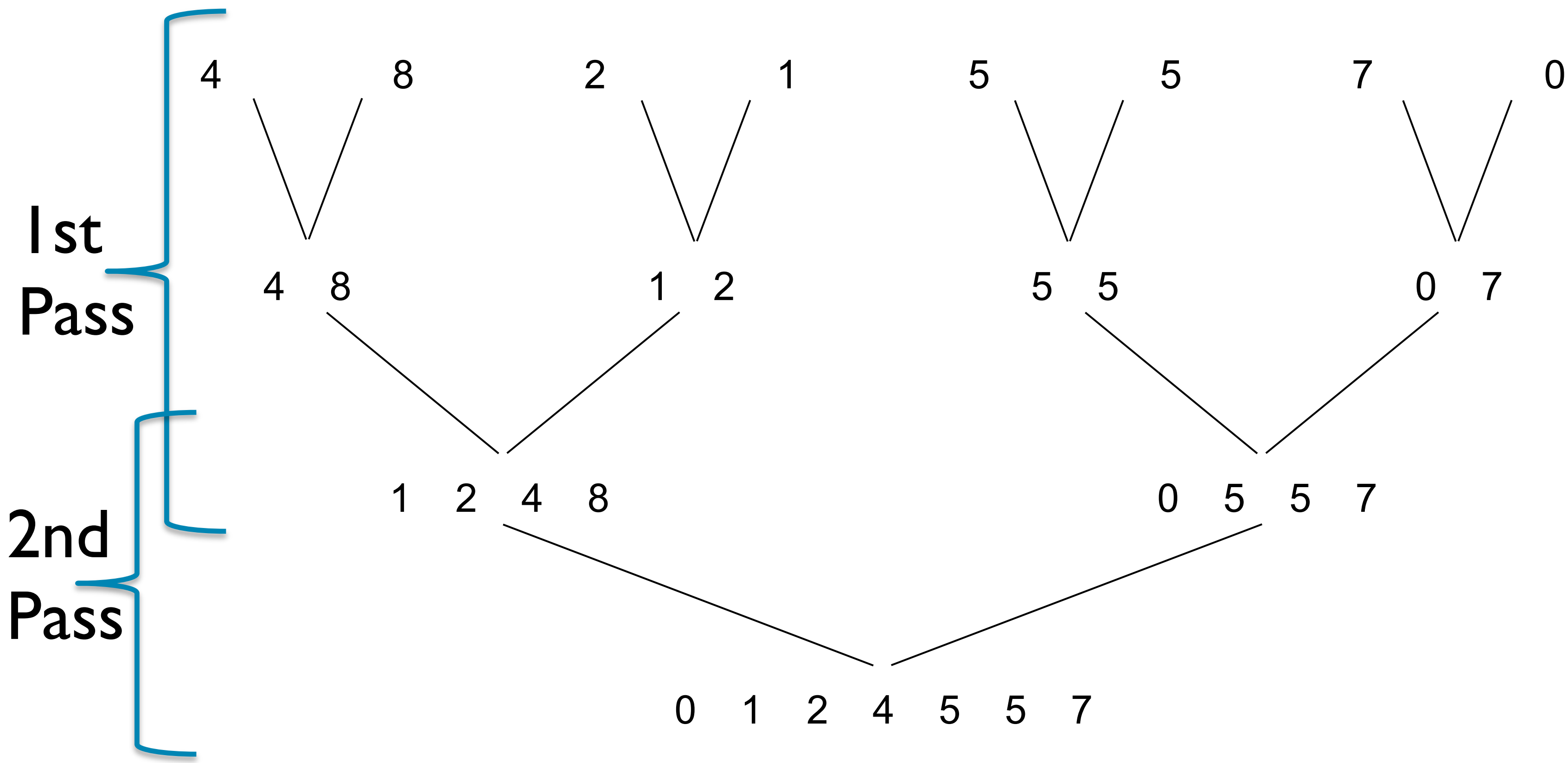
- Scan two sorted columns, output matching values
  - Can have associated values or record IDs
  - Output cross product when multiple values
  - Generally viewed as the "free" thing after sorting
    - More an indication of how slow sorting is
- Software implementations have bad branching behaviour
  - Limits the IPC → hard to keep up with memory

Merge Join

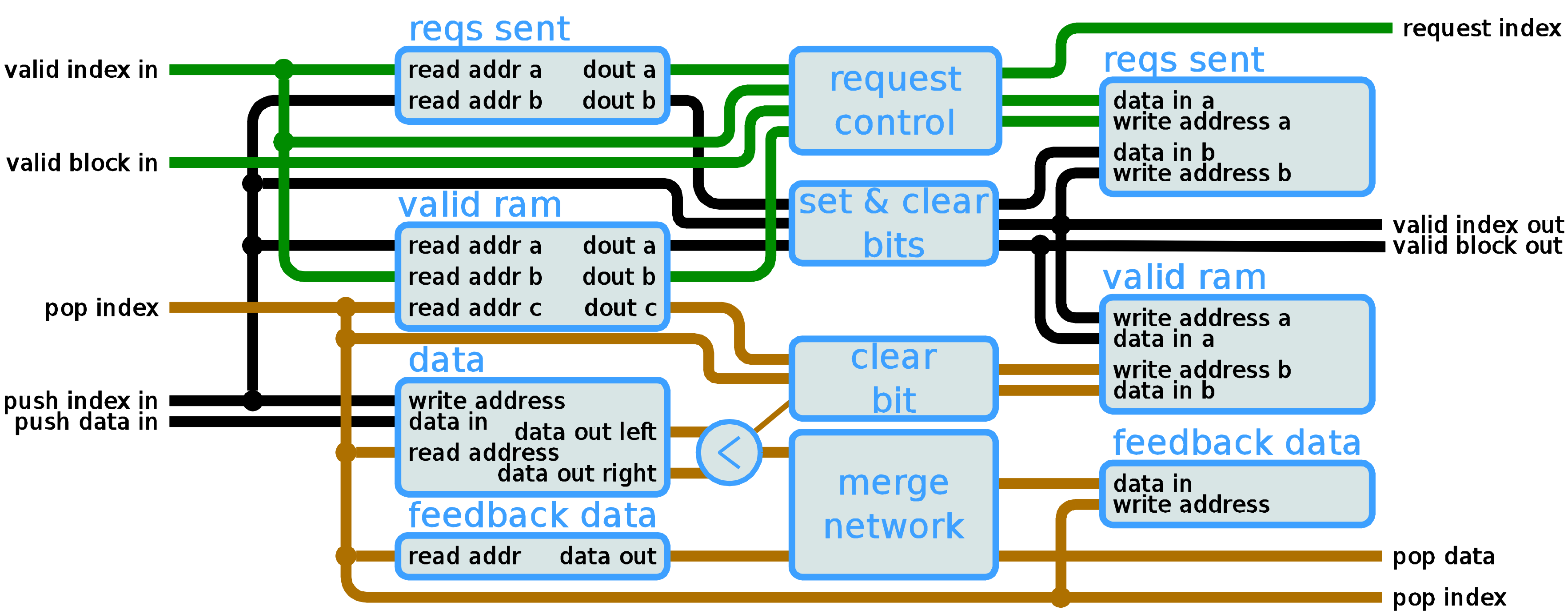


- Output is bitmask of equal keys with corresponding values
  - Ready for input into the select block

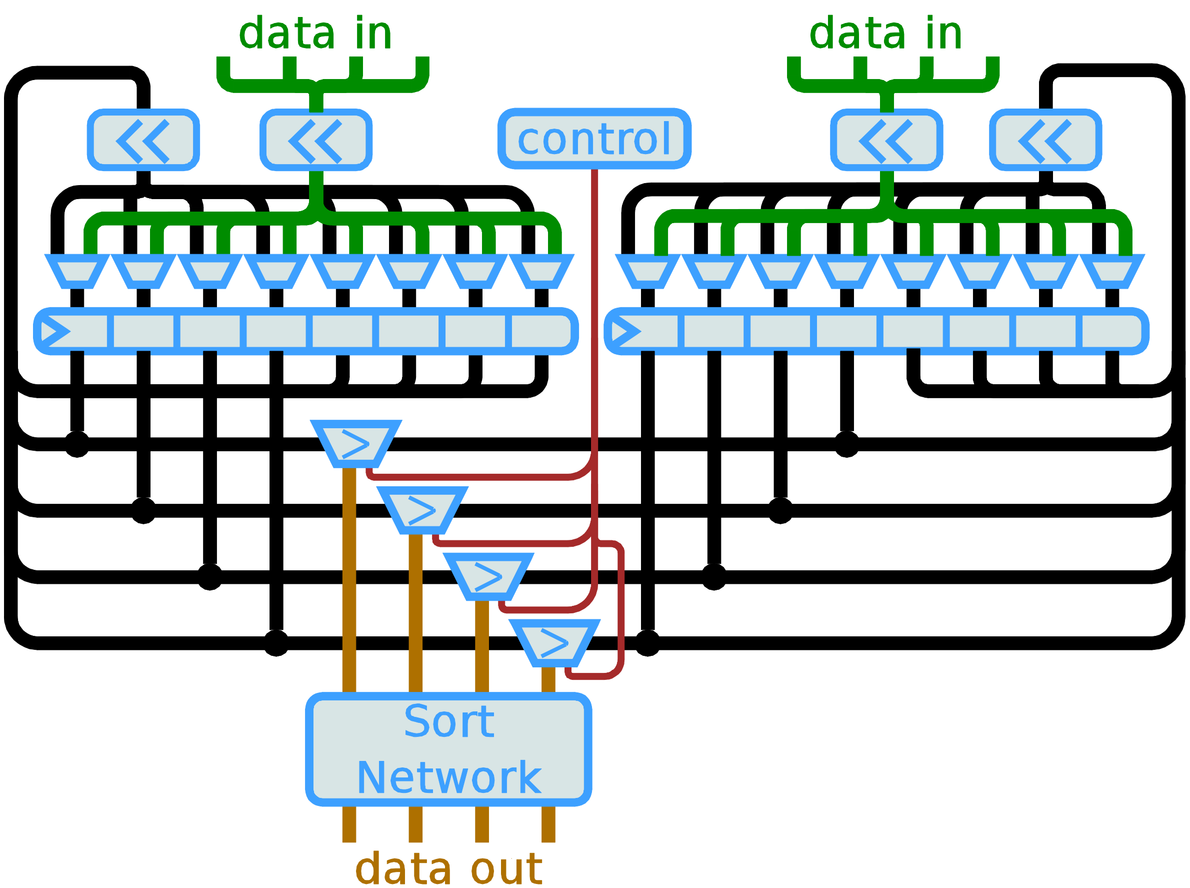
Merge Sort



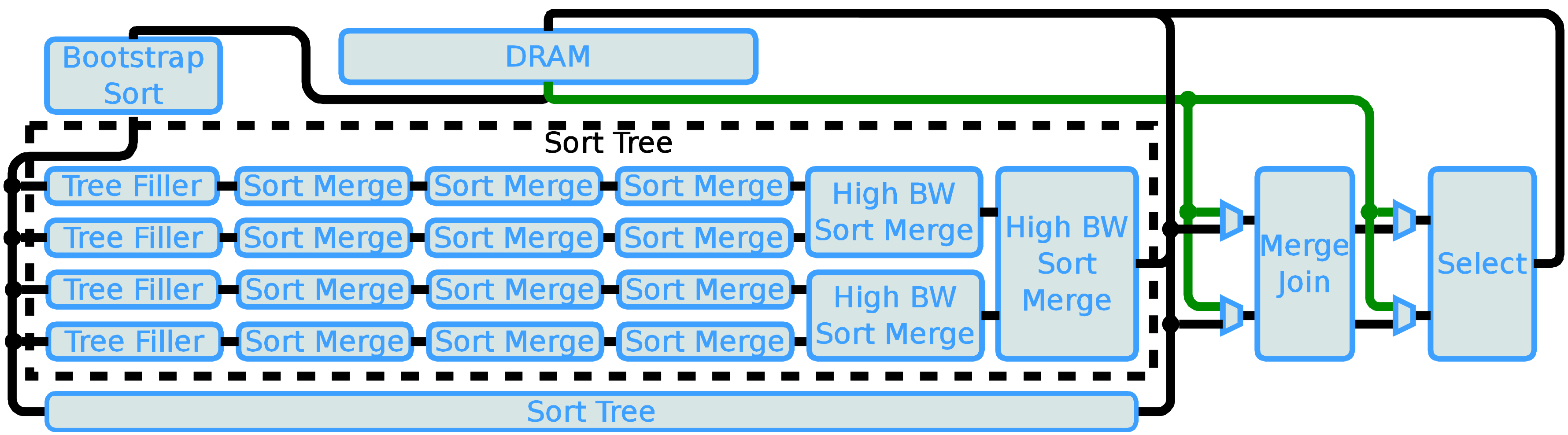
Merge Sort Level



High Bandwidth Sort Merge Node

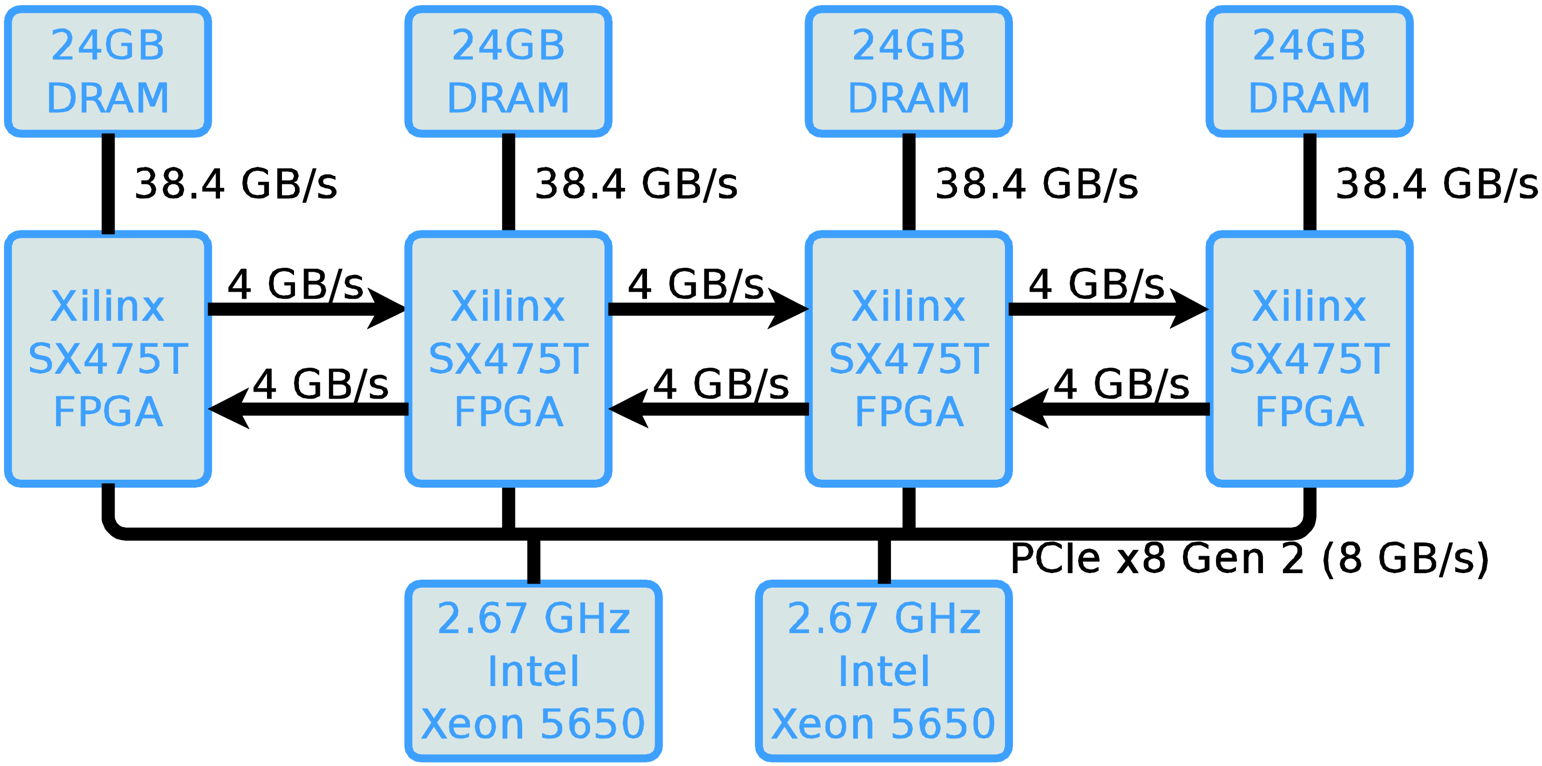


Sort Merge Join

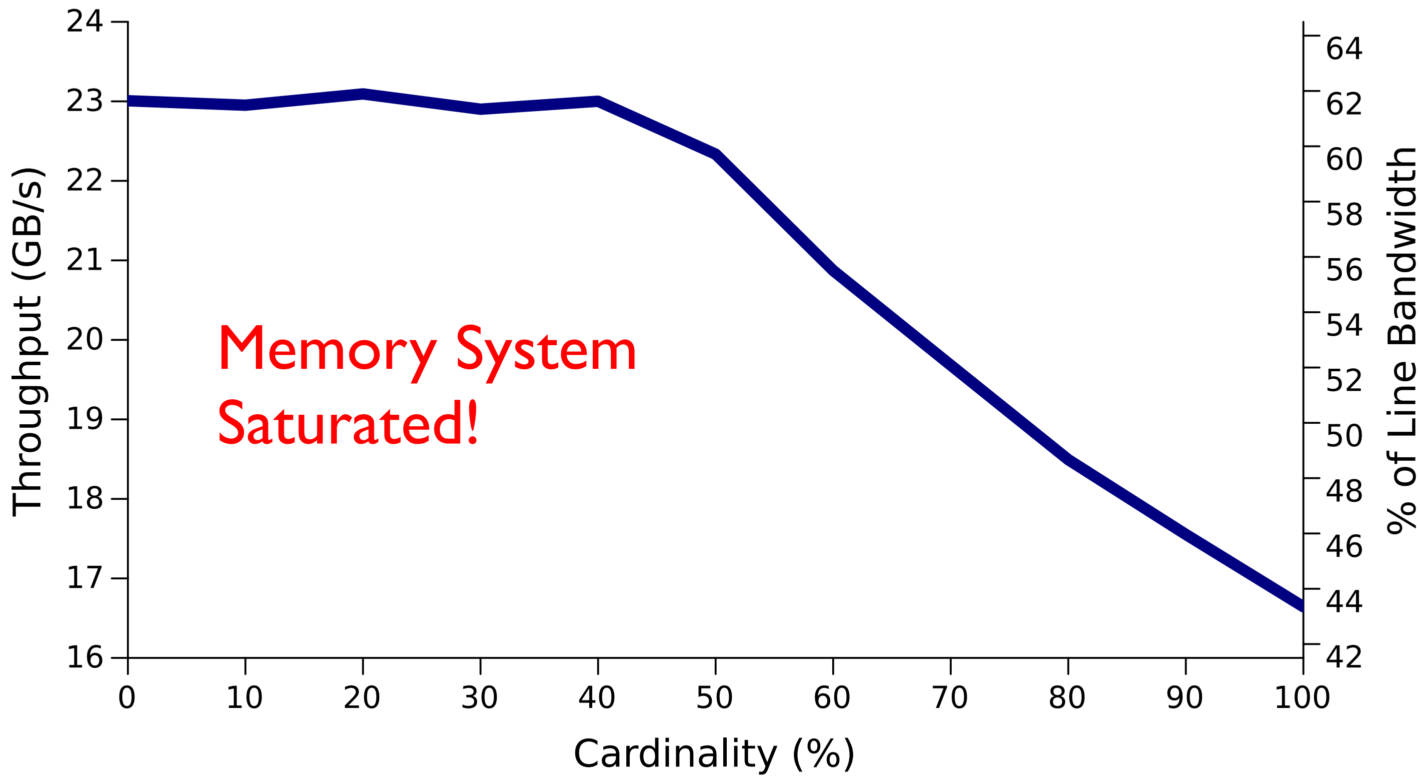


- Sort, merge join, and select blocks are combined to perform an full sort merge join in hardware

Prototyping Platform - Maxeler

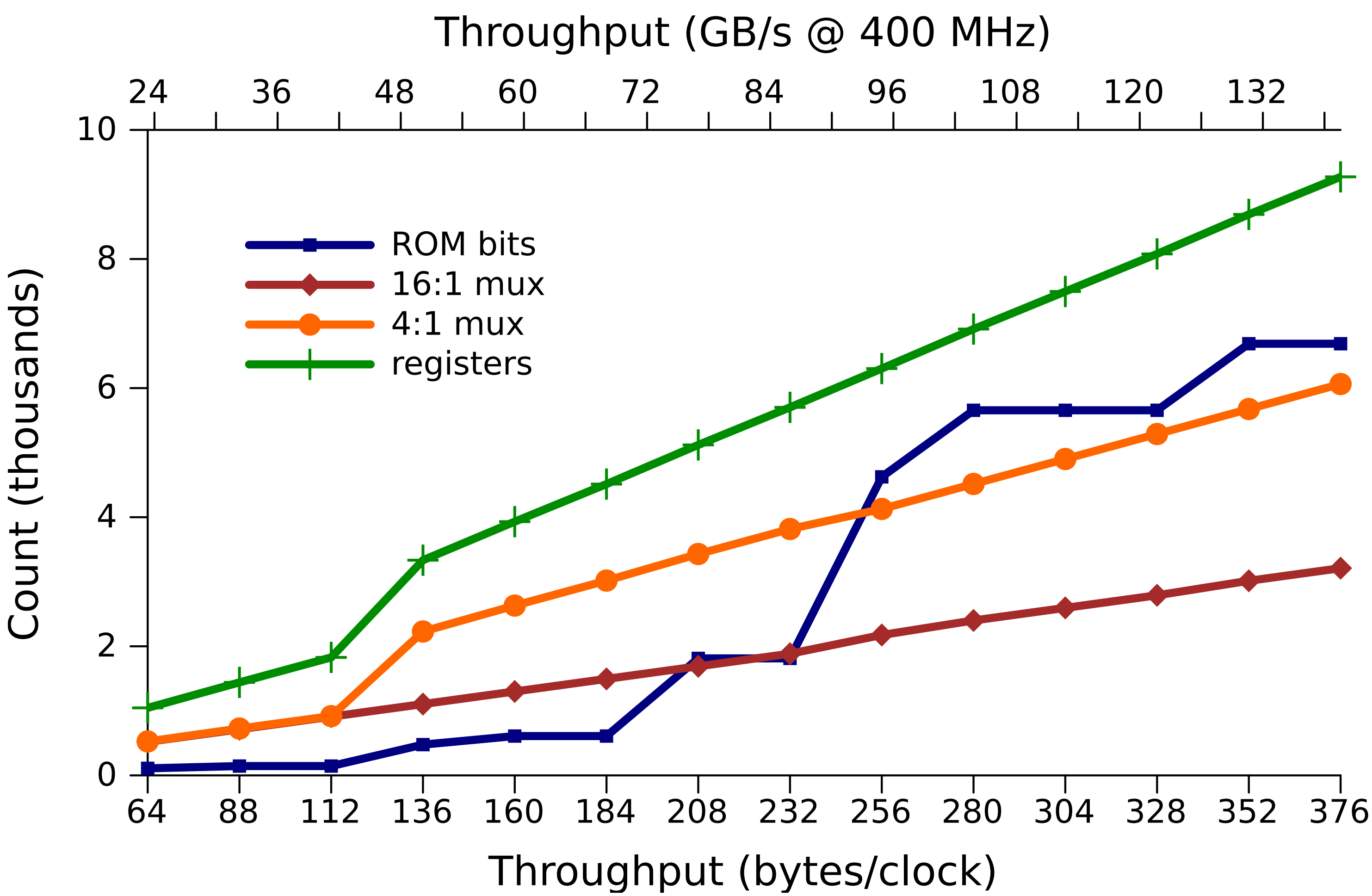


Select Throughput

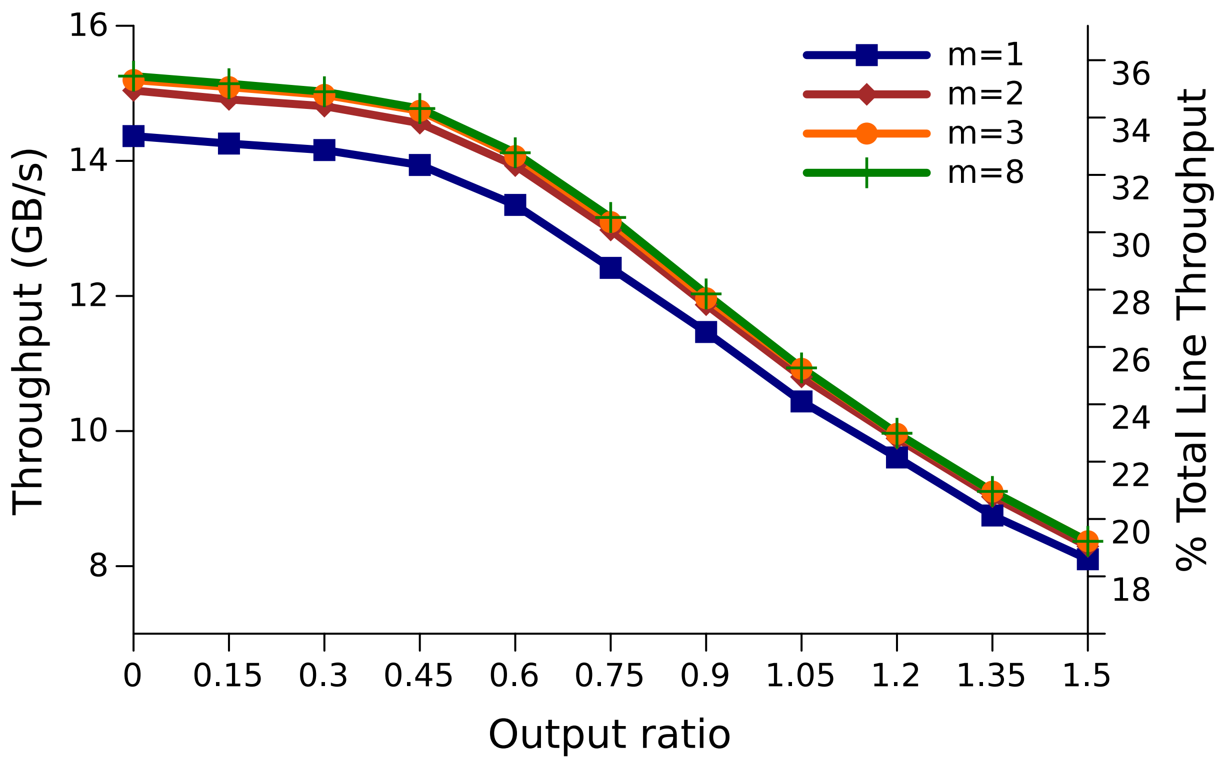


- Software achieved 7 GB/s (33%)
  - STREAM achieved 12 GB/s (57%)

Select Resources

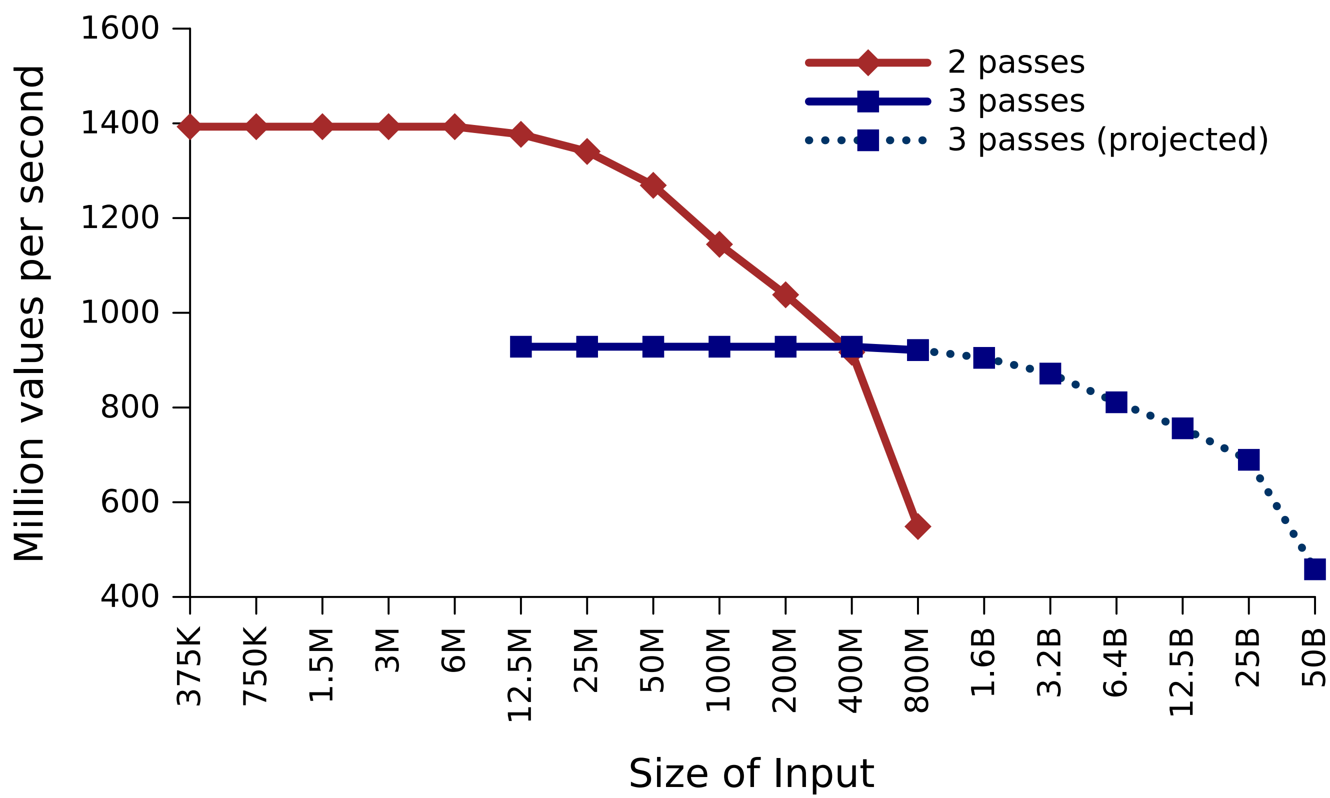


Merge Join Throughput



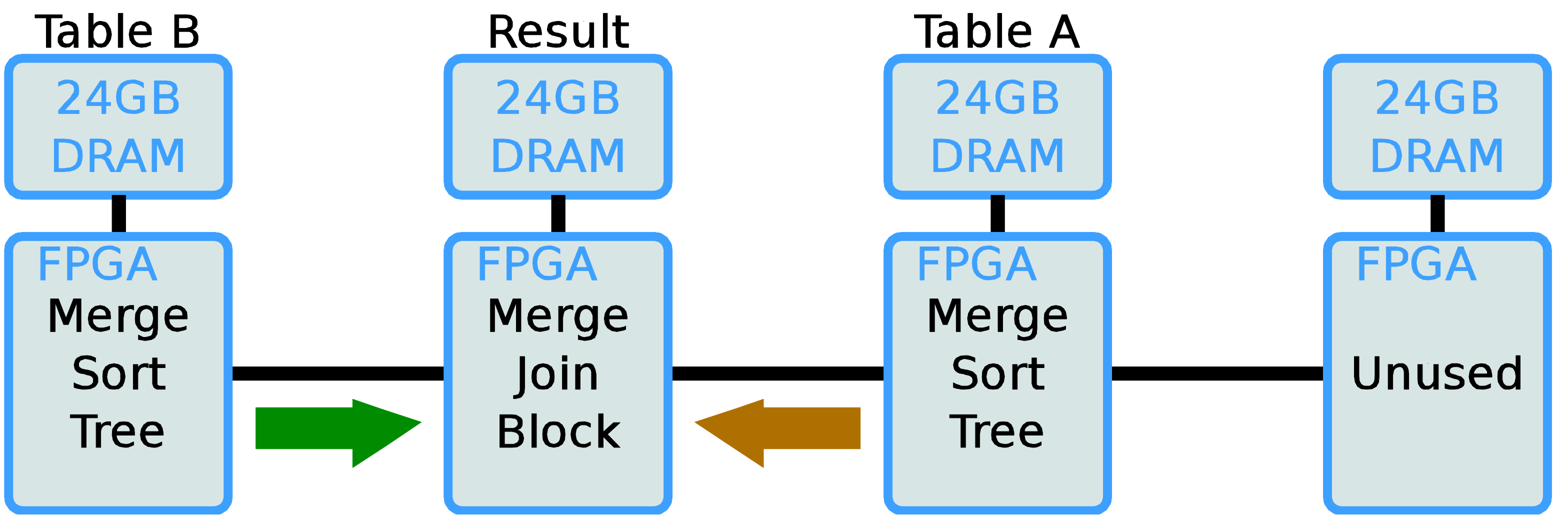
- Resources required is a quadratic function of desired bandwidth
  - All in comparison logic, routing was the limiting factor
- Above 1.5x output, write bandwidth dominates
  - Throughput above is input consumed

Sort throughput



- Resources required is a linear function of desired input size
  - Dominated by the memory required to hold working sets
- Recent CPU/GPU numbers ~300M 32-bit values per second

Sort Merge Join



- Performance limited by intra-FPGA link
- Total throughput is 800 million tuples/second
  - ~6.5 GB/s
  - 8x previous work on software joins

Conclusions

- FPGAs can be used to saturate memory bandwidth in ways that processors can not
  - Make the most of every byte read
  - In some cases, address bandwidth is just as important as raw data bandwidth
- Scaling your design to high bandwidths can greatly influence the architecture
  - Think streaming
- Next step is to interact with the rest of the system

Questions?