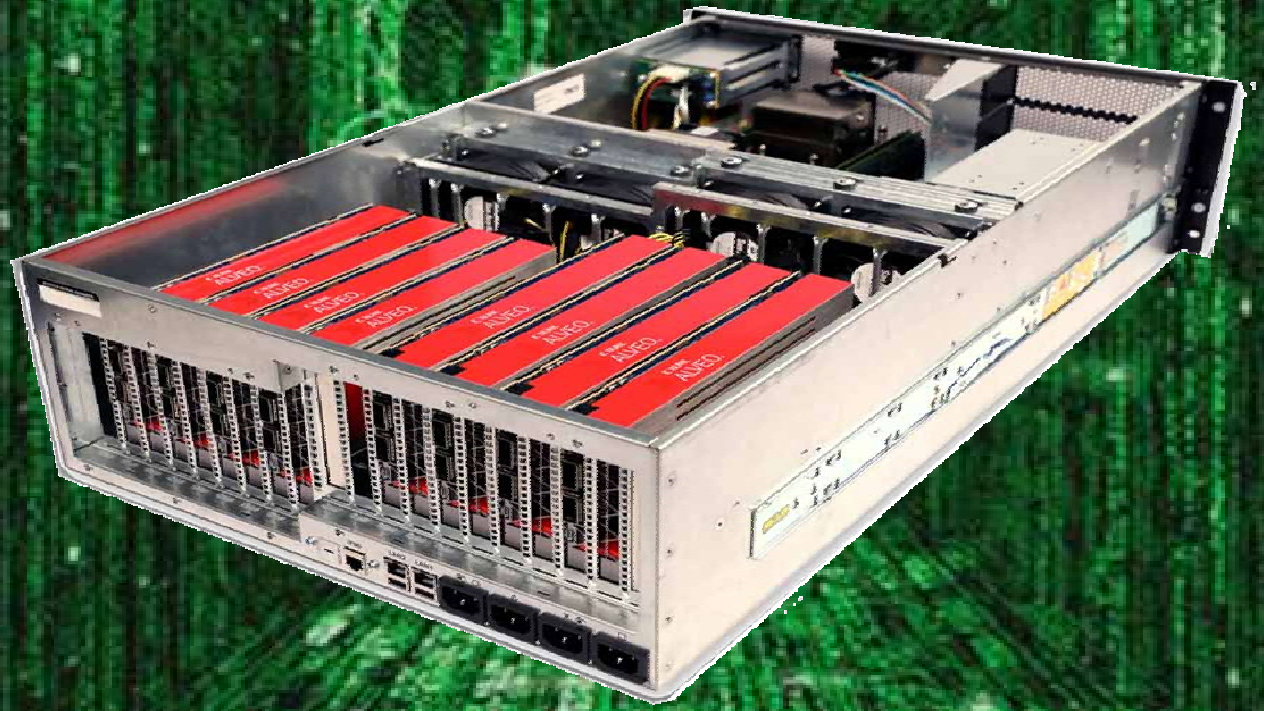
Better Performance through Resource-Elastic Dynamic Stream Processing on FPGAs



Dirk Koch, dirk.koch@ziti.uni-heidelberg.de
Heidelberg University (Germany), The University of Manchester (UK)

Dynamic FPGA Acceleration in Data Centers

- **Security**

  (important but not the core of this talk)

- **"Plumbing"**

  (How to implement infrastructure and user modules)

- **Runtimes and Services**

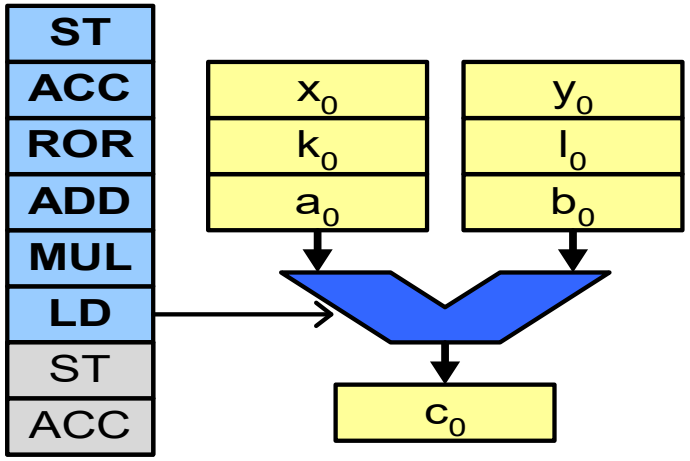
  (APIs, programming models, drivers,…)

- **Applications**

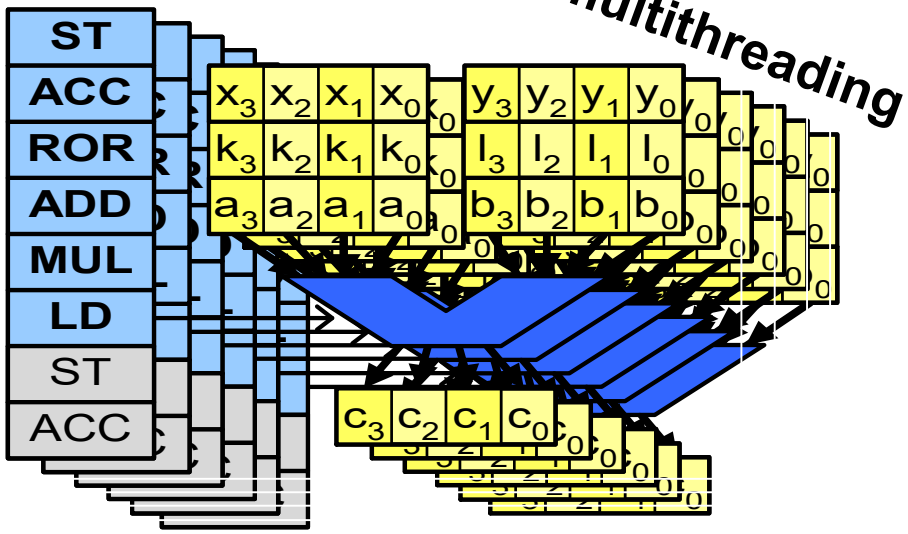
  (data analytics and database acceleration)

CPUs vs. GPUs vs. FPGAs

Scalar processing

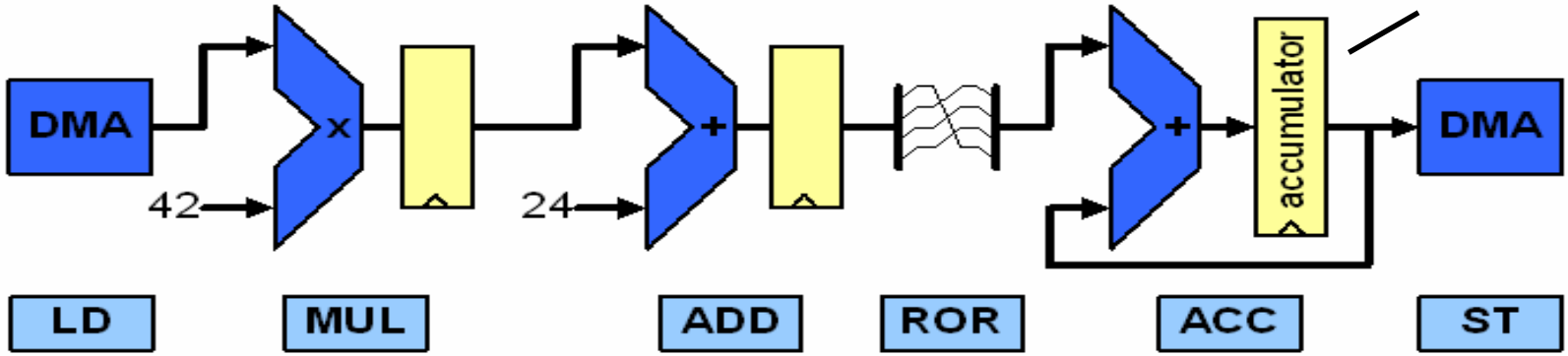


SIMD processing (e.g. GPU)



Dataflow processing / pipelining (FPGAs)

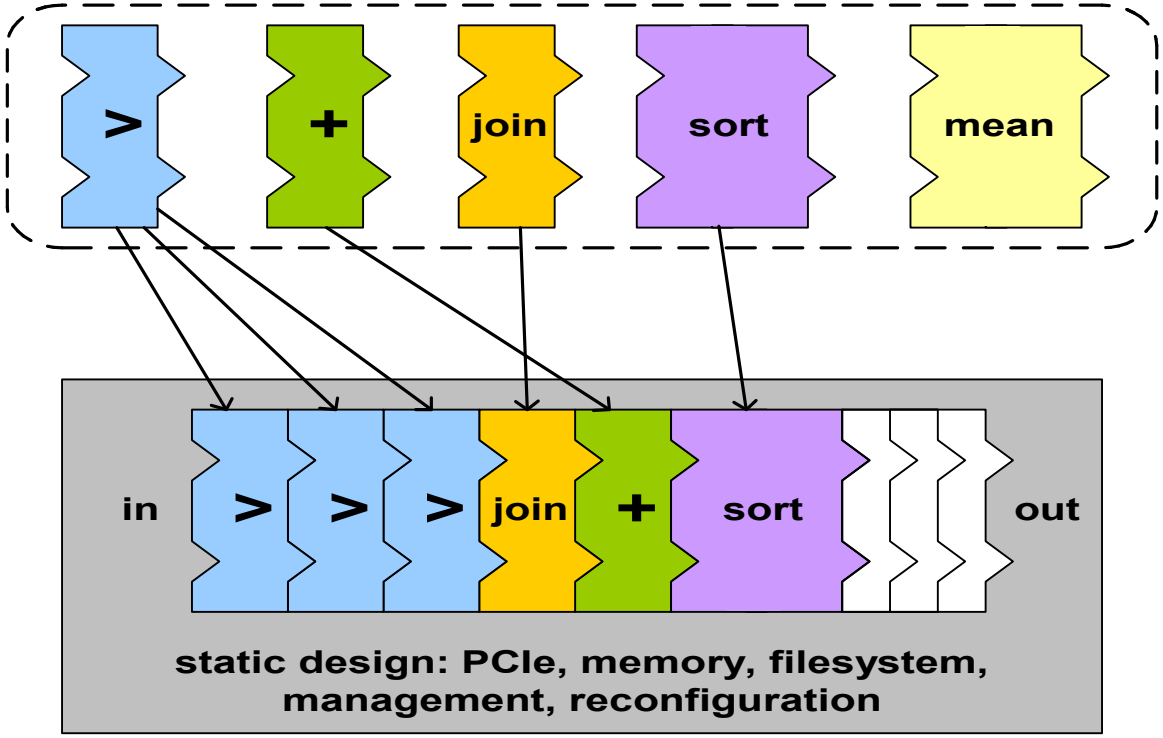
Pipelining often better for global states



FPGAs are more than updatable ASICs!

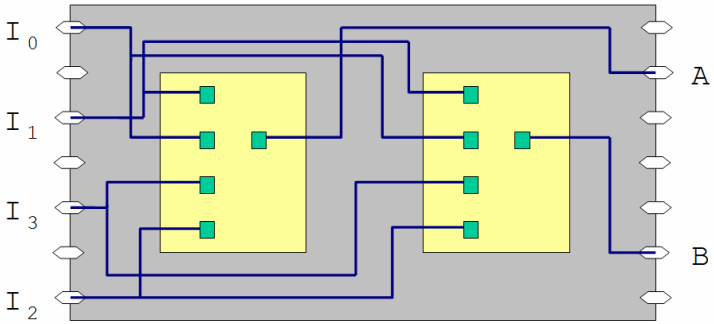
Database acceleration example

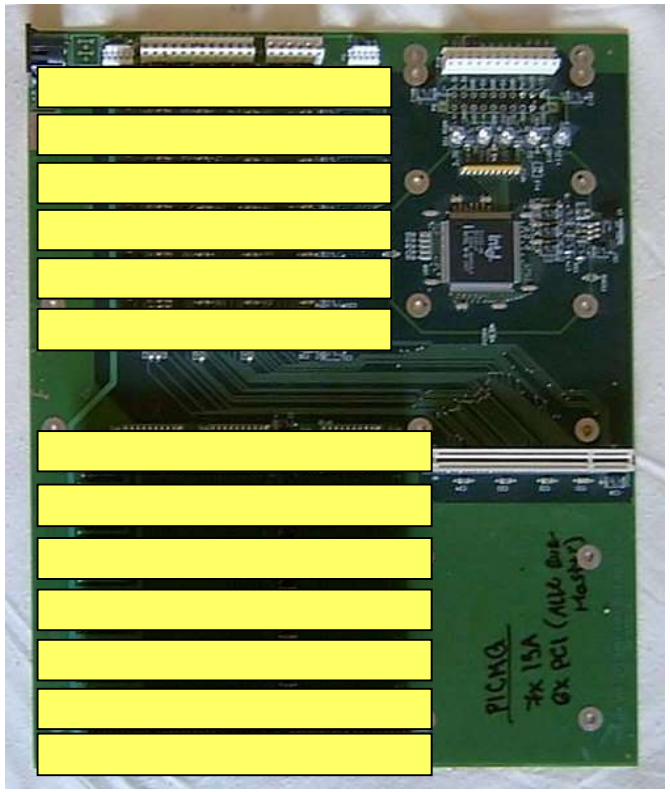
- ASIC-like (fixed) accelerators often require an over-provisioning (poor resource usage)
- Better: use FPGA reconfiguration to just load currently needed modules →provides more resources to the currently running tasks (faster!)



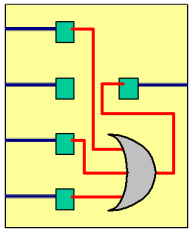
- What about having a database or data analytics system where we can plug together operator modules to solve problems that we may only know at runtime?
  → **Dynamic Stream Processing** (uses partial reconfiguration)

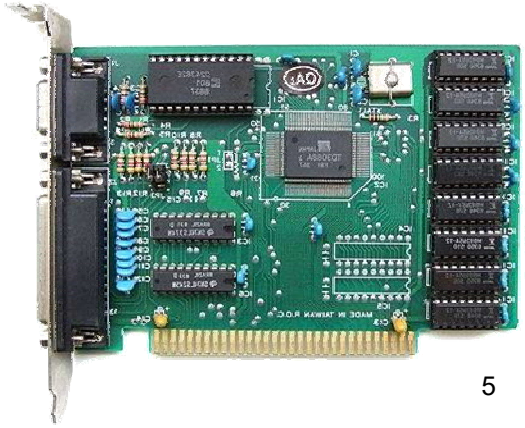
GoAhead PR Design Flow



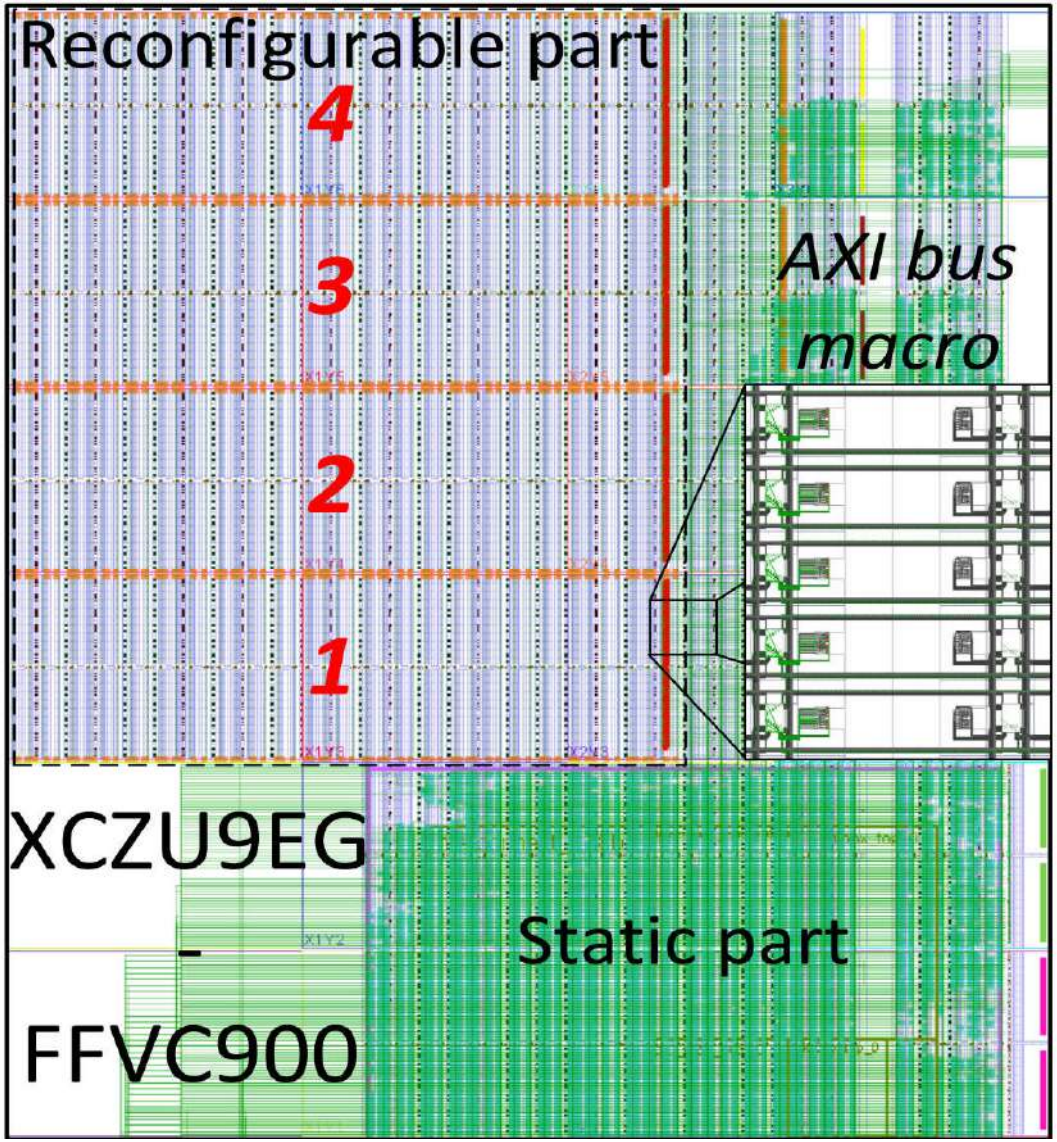


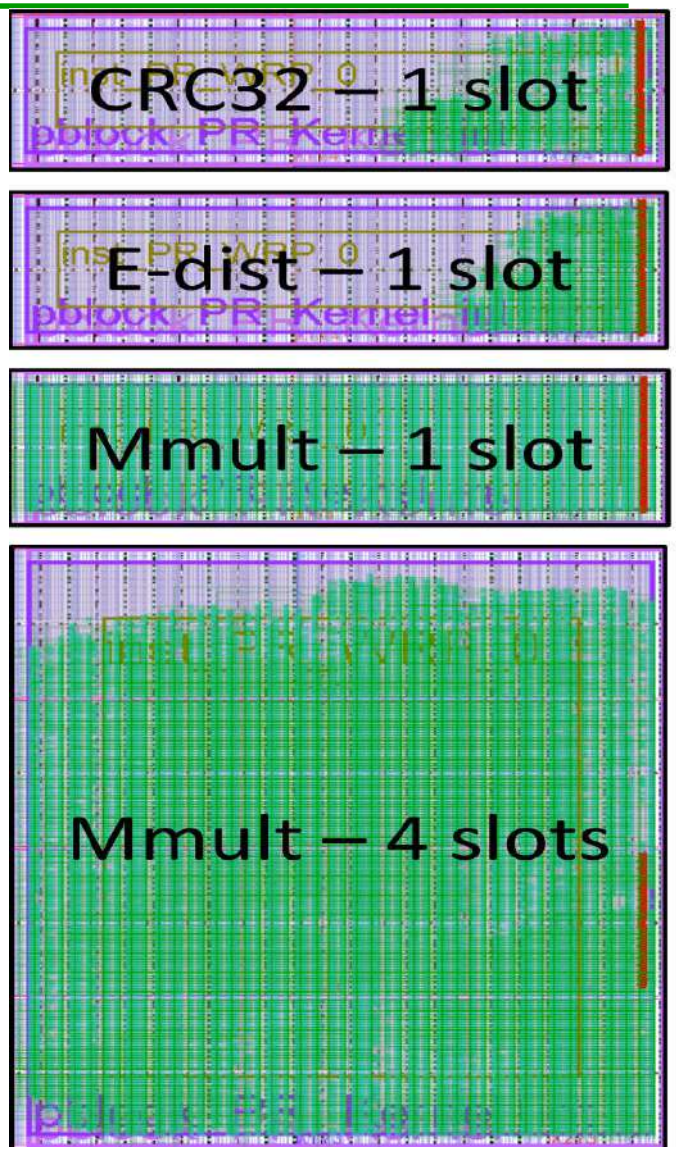
- The partial regions in a static system correspond to (ISA/PCI) slots on a PCB
- Modules are the equivalent to hot-swappable cards
- Properties:
  - Static system and modules implemented independently
  - Simple integration through netlist or bitstream linking (including partial reconfiguration)
  - Arbitrarily interchangeable



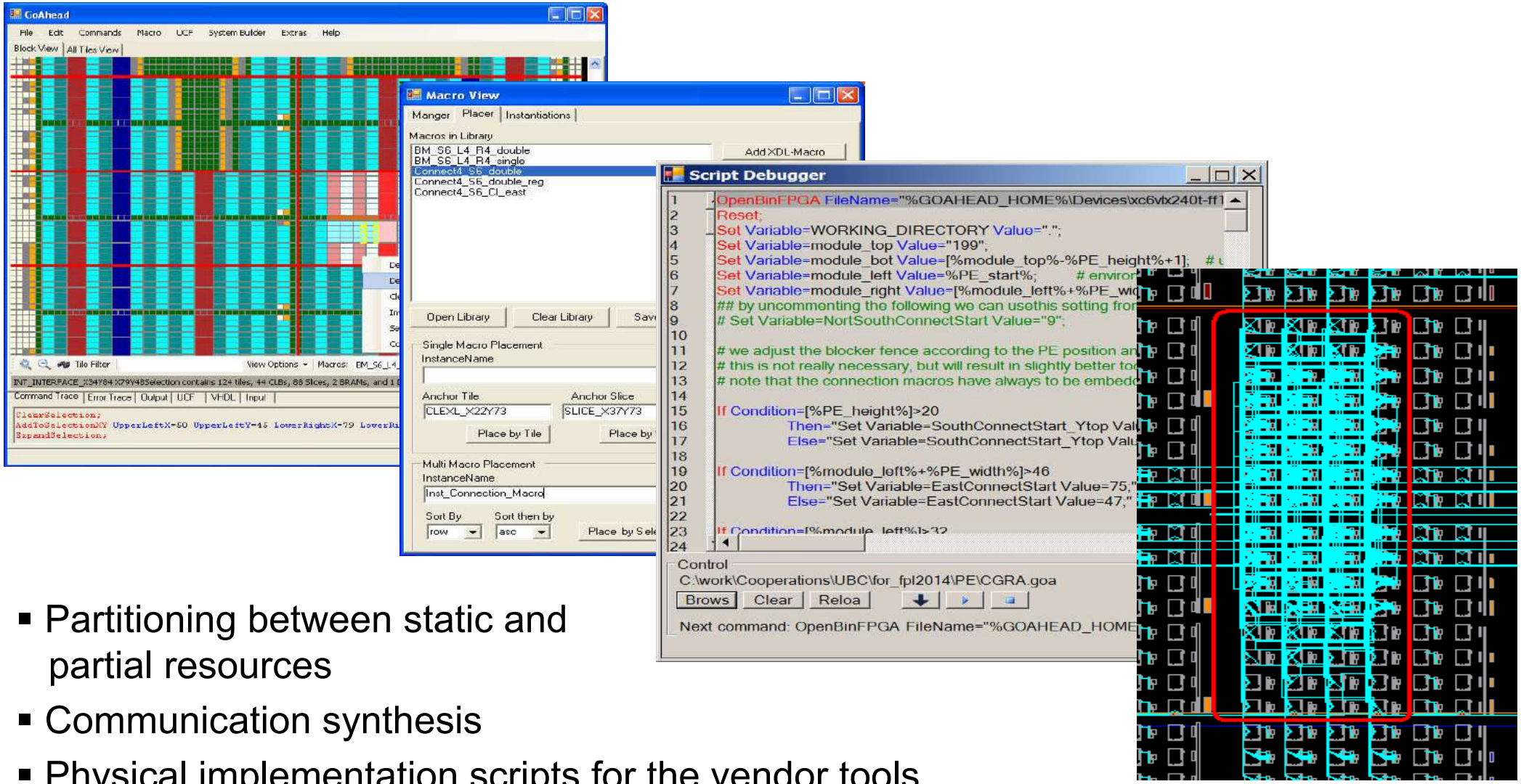


ZUCL (running OpenCL on Zynq UltraScale+, FSP'18)



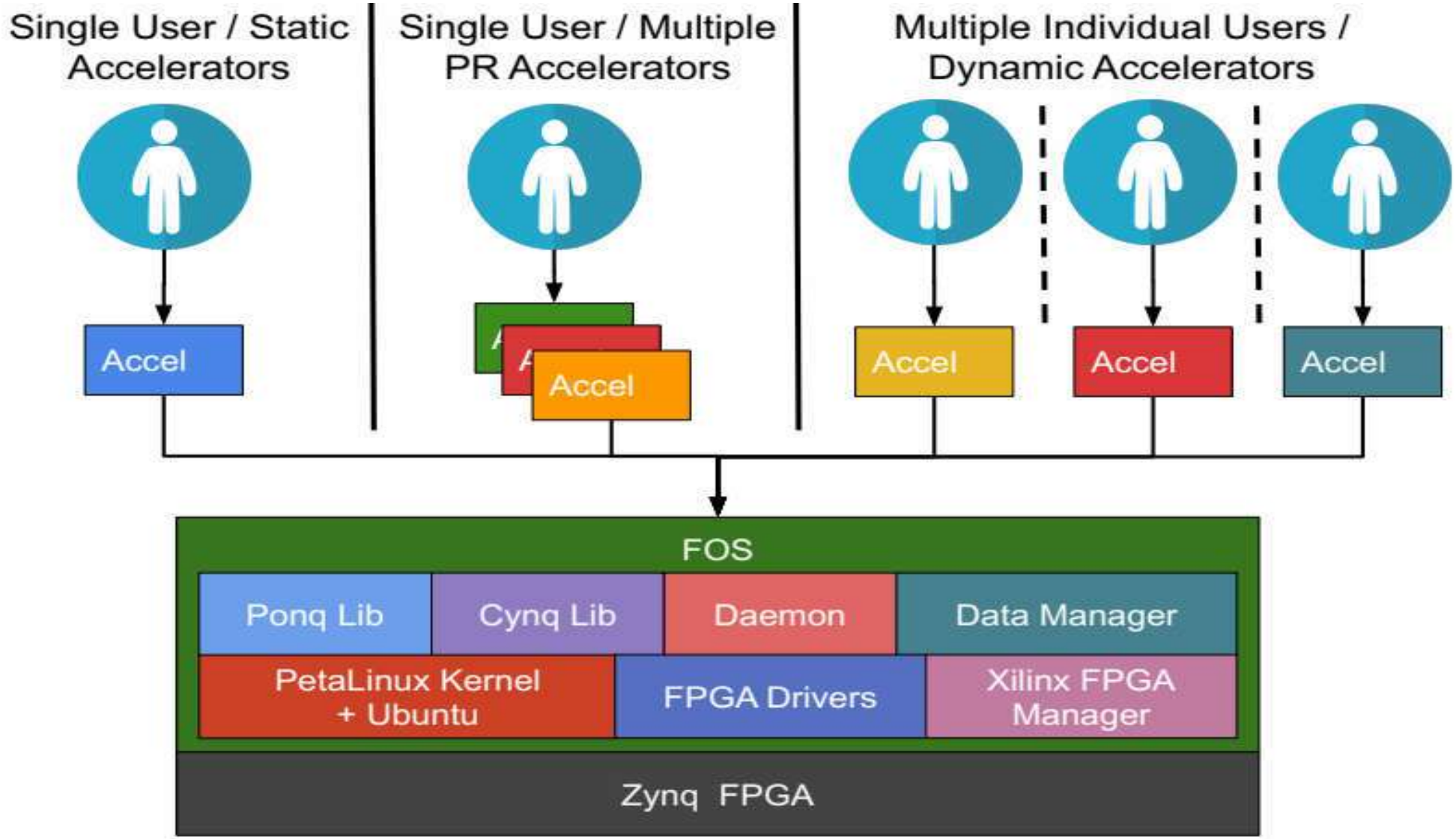


GoAhead: **a tool for implementing partially reconfigurable systems**



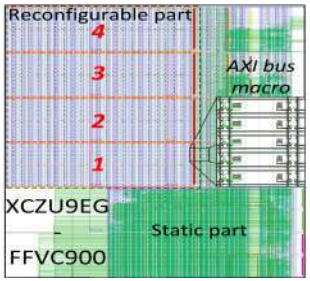
- Partitioning between static and partial resources
- Communication synthesis
- Physical implementation scripts for the vendor tools

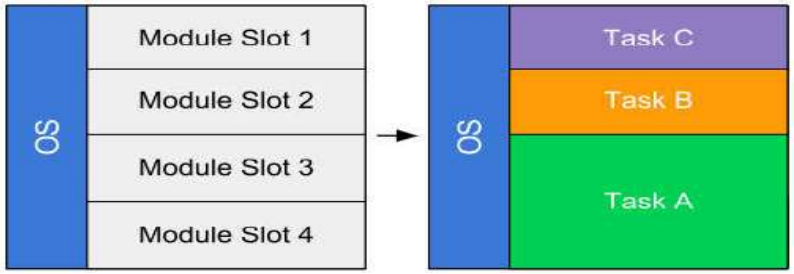
The FOS (FPGA Operating System) Stack

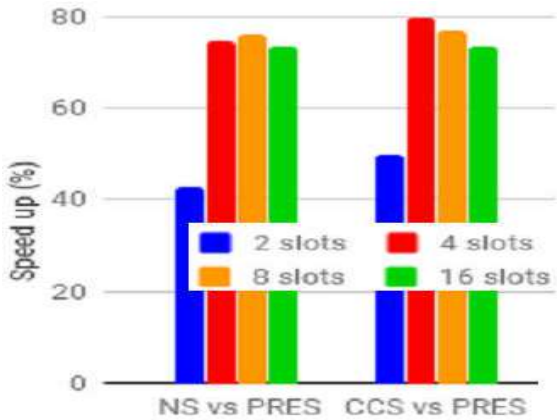


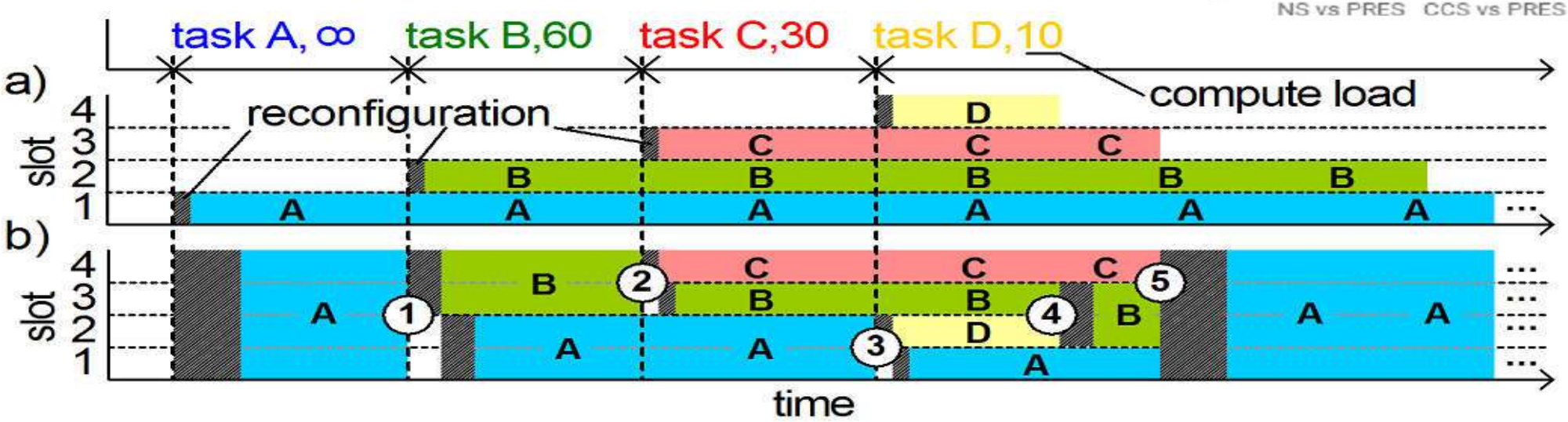
Resource Elastic Virtualization on FOS

- **Resource Elastic Virtualization for FPGAs for OpenCL**
  - **Using aggressive reconfiguration to keep utilization high in a dynamic scenario**
  - **Virtualization in the space-domain (time-domain fallback, if needed)**



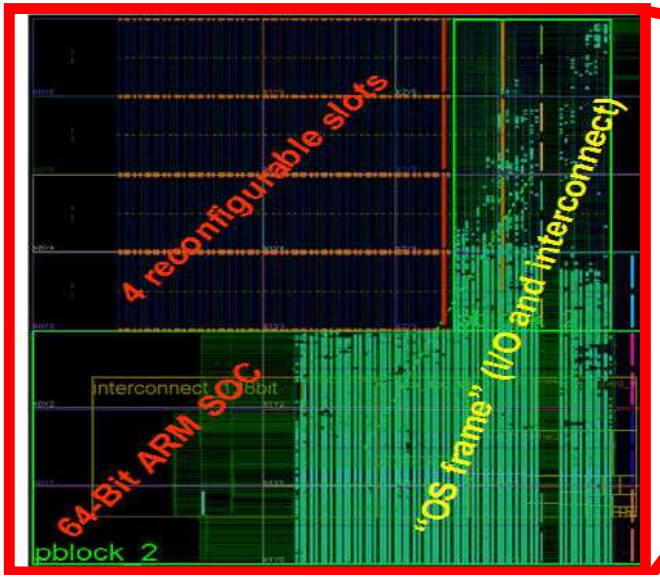


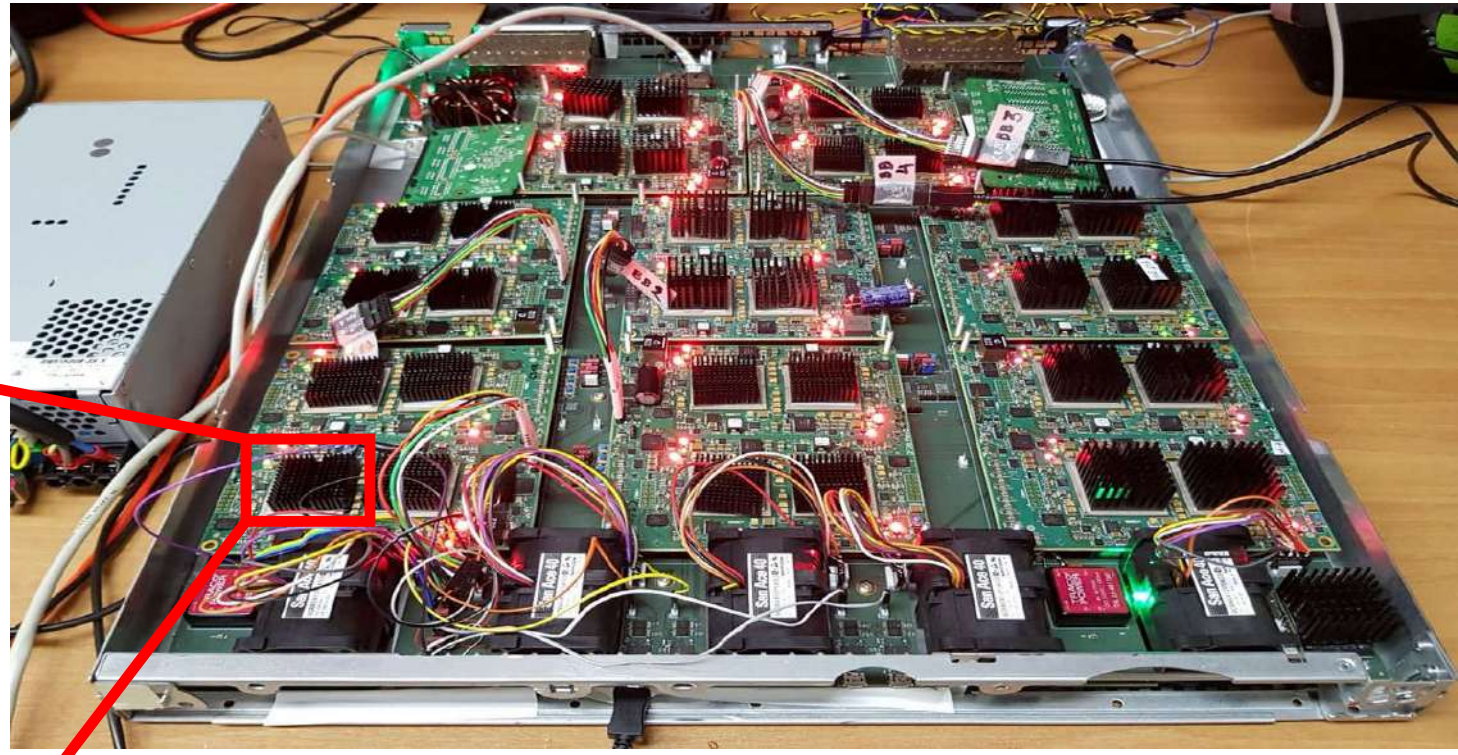




FPGAs for Datacenters (H2020 ECOSCALE)

ECOSCALE demonstrator fully-populated 1u blade with
32 x Zynq UltraScale+ (ZU9EG with 16GB/FPGA or 512 GB/blade)

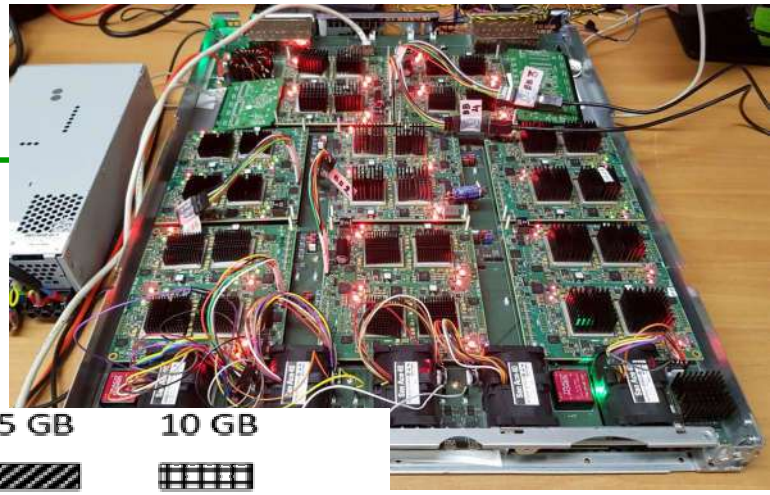


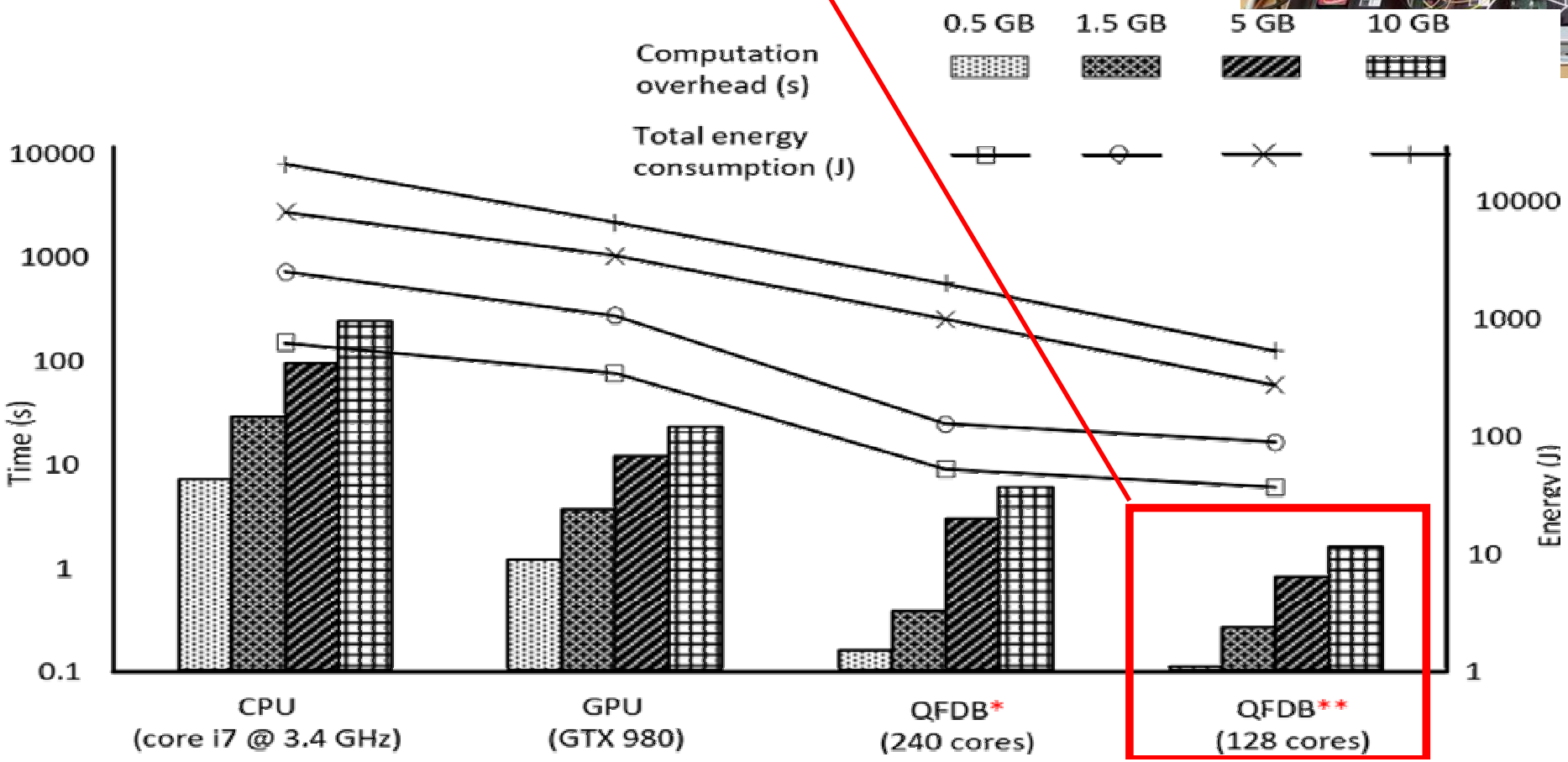


www.ecoscale.eu

FPGAs for Datacenters

Partial reconfiguration allows moving compute to data → huge energy savings

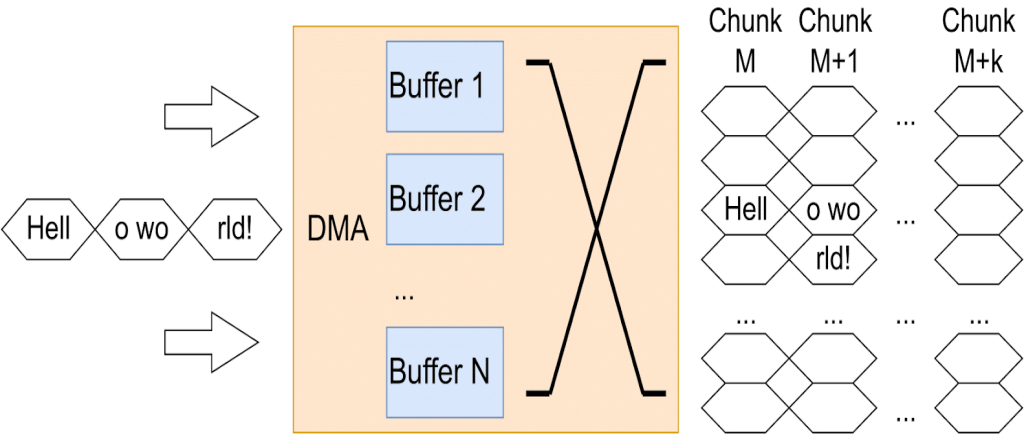


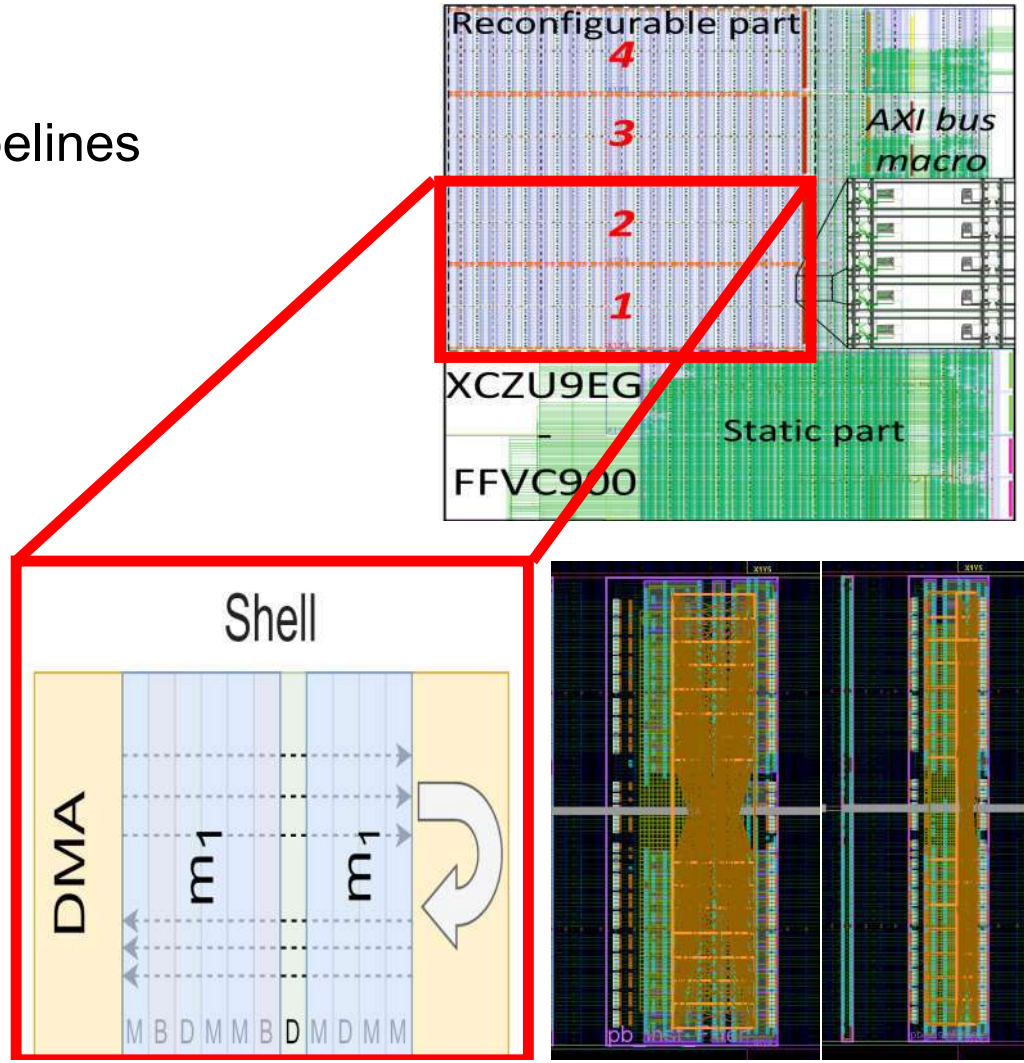


Dynamic Stream Processing

Integrated into FOS

- Run monolythic accelerators **and...**
- ...Run composable stream processing pipelines → supported by a DMA engine and dedicated communication protocol
  - Virtual streams
  - Credit-based flow control
  - Register-file read/write access

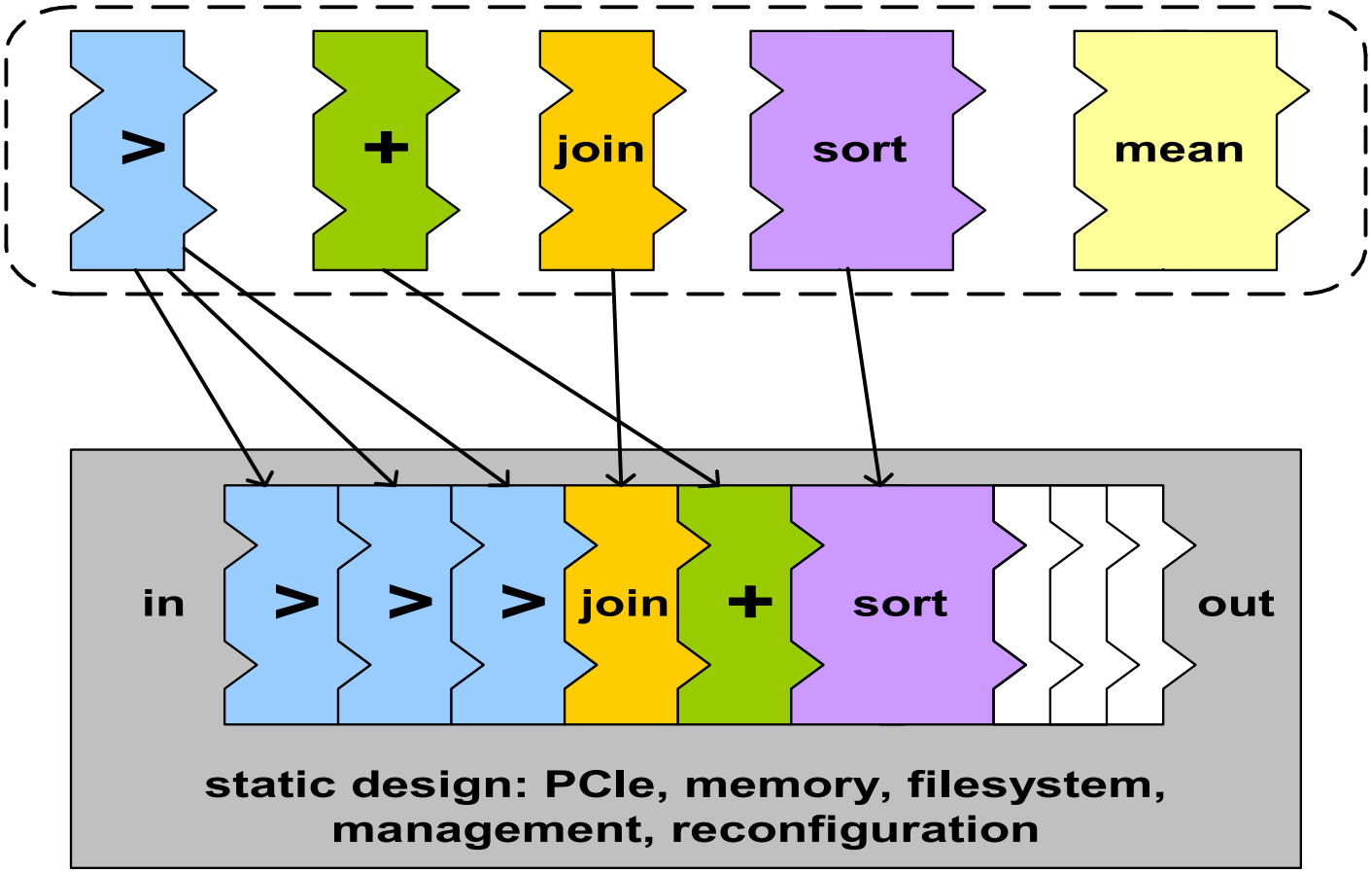




Dynamic Stream Processing

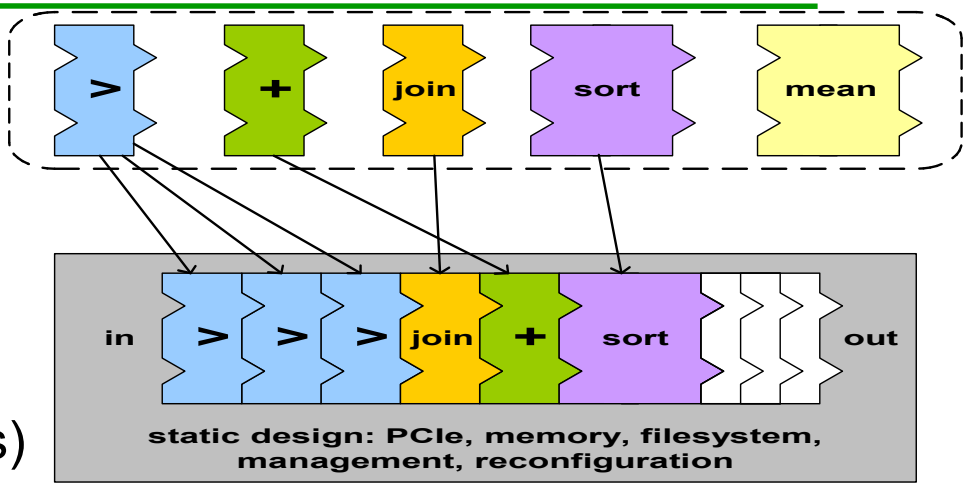
Database acceleration example

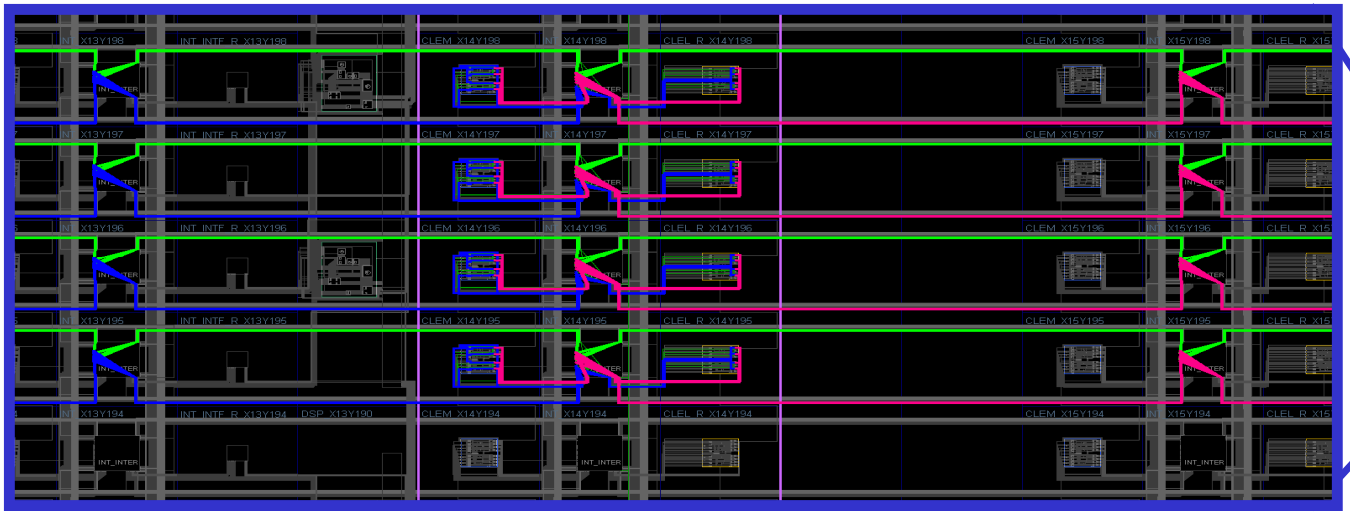
- Build library with SQL operators
- Compose optimized datapath at run-time
- Blocking operators are natural reconfiguration point!
  → operators before and after a sorter run mutual exclusive (never together)

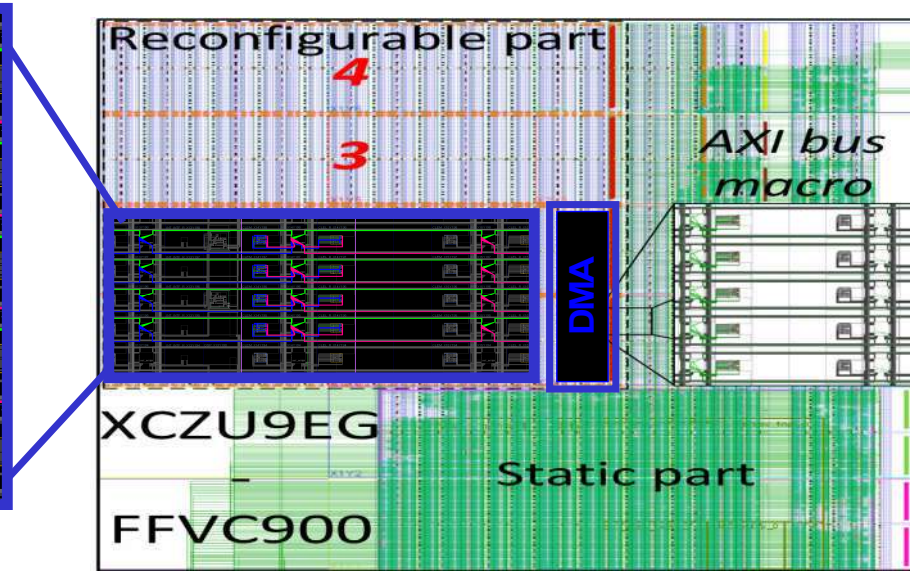


Dynamic Stream Processing

- Can run all TCP-H queries
- Modules with different area-performance/utility tradeoffs
- Modules with different resource footprints (improves placement for heterogeneous resources)



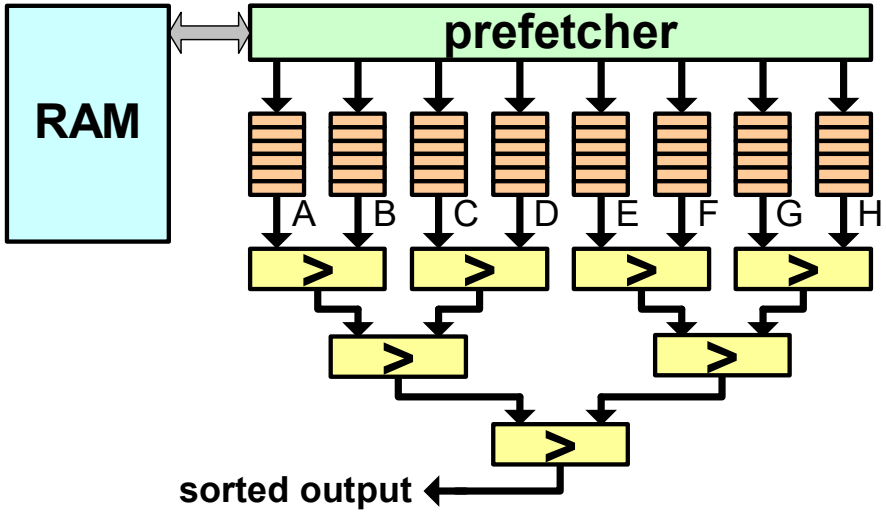


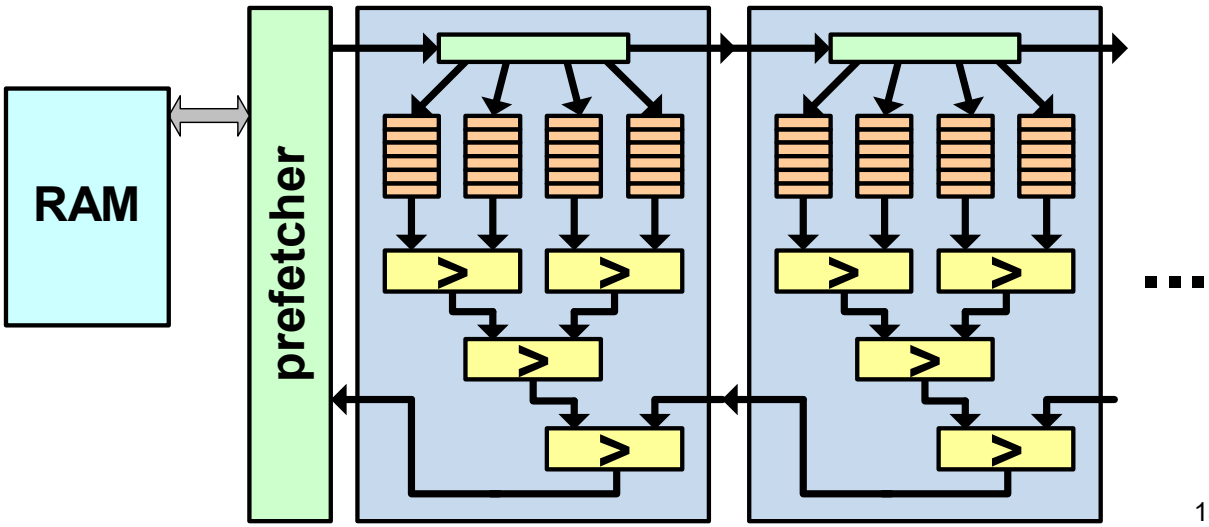


Dynamic Stream Processing – Resource Elasticity

Resource Elasticity allows a runtime system to maximize throughput/utility for the currently available resources.

- Example Sorting: build larger sorters from composable accelerator PEs → more work-per-run → fewer runs (High-utility sorting can merge thosands of streams in one run!)
- Works for WHERE clauses (number and complexity of clauses/regular expressions)
- JOIN: larger buffering reduces runtime



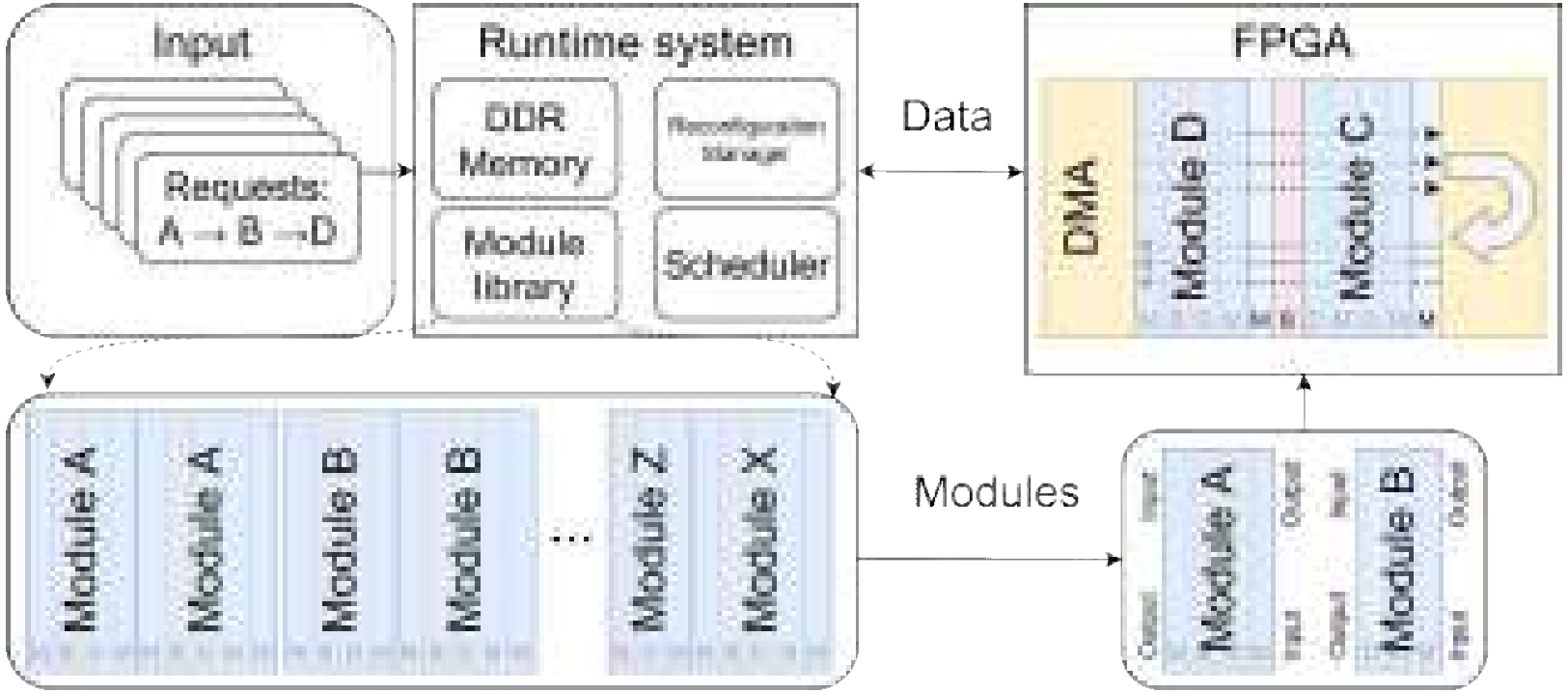


```
SELECT Sum(l_extendedprice * ( 1 - l_discount )) AS revenue
FROM   lineitem,
       part
WHERE  ( p_partkey = l_partkey
         AND p_brand = 'Brand#12'
         AND p_container IN ( 'SM CASE', 'SM BOX', 'SM PACK', 'SM PKG' )
         AND l_quantity >= 1
         AND l_quantity <= 1 + 10
         AND p_size BETWEEN 1 AND 5
         AND l_shipmode IN ( 'AIR', 'AIR REG' )
         AND l_shipinstruct = 'DELIVER IN PERSON' )
        OR ( p_partkey = l_partkey
             AND p_brand = 'Brand#23'
             AND p_container IN ( 'MED BAG', 'MED BOX', 'MED PKG', 'MED PACK' )
             AND l_quantity >= 10
             AND l_quantity <= 10 + 10
             AND p_size BETWEEN 1 AND 10
             AND l_shipmode IN ( 'AIR', 'AIR REG' )
             AND l_shipinstruct = 'DELIVER IN PERSON' )
        OR ( p_partkey = l_partkey
             AND p_brand = 'Brand#34'
             AND p_container IN ( 'LG CASE', 'LG BOX', 'LG PACK', 'LG PKG' )
             AND l_quantity >= 20
             AND l_quantity <= 20 + 10
             AND p_size BETWEEN 1 AND 15
             AND l_shipmode IN ( 'AIR', 'AIR REG' )
             AND l_shipinstruct = 'DELIVER IN PERSON' );
```

We can execute TCP-H Query 19, Xilinx cannot (Vivado HLS) ☺

Stream Processing Ecosystem

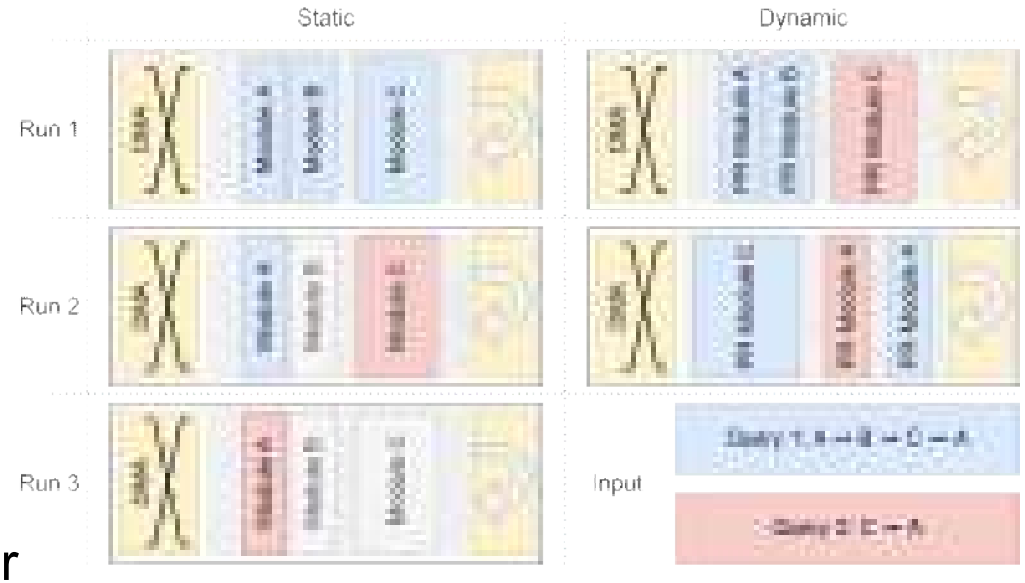
- Frontend
  - Parser
- Middleware
  - Scheduler
  - Operation management
- Hardware
  - Static
  - Module library (including variants for different placement positions and area-performance tradeoffs)

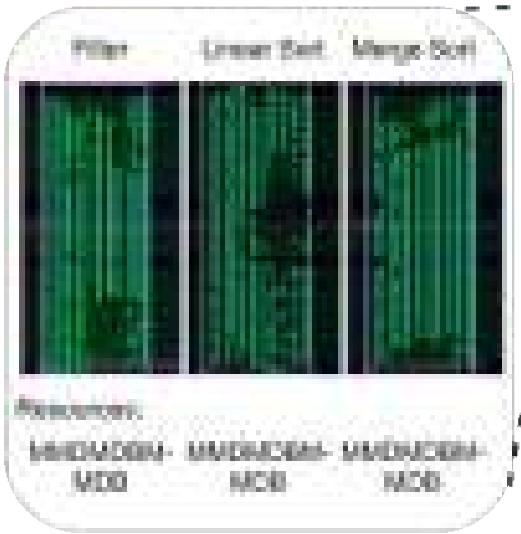


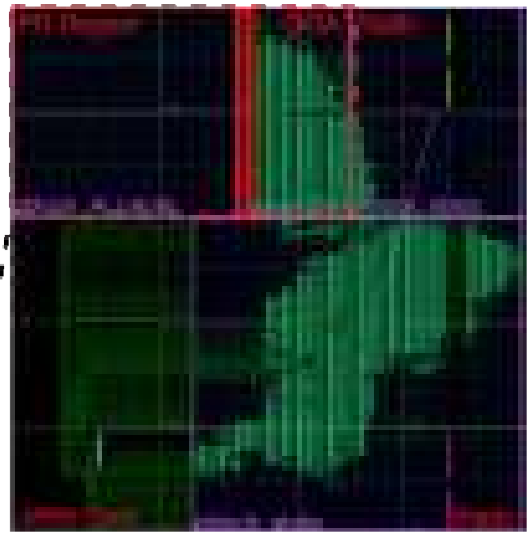
Dynamic Stream Processing

Integrated into FOS

- Frontend:
  - Parse SQL – PostgreSQL
  - Create execution graph
- Middleware:
  - Schedule modules - Branch and Bound
  - Configure FPGA – Xilinx FPGA manager
  - Initialise modules with generic drivers
- Hardware:
  - Execute
  - Collect results







Resource-Elastic Dynamic Stream Processing

- FPGA acceleration allows us to tailor the I/O and memory subsystem to our problems (e.g., we can sort thousands of streams in a single run, thereby minimizing #runs) →design pattern fits to many big data stream processing problems and allows us to maximize the compute work per unit I/O (I/O is key for low power and for everything else)
- Our dynamic stream processing was demonstrated for SQL (and additional data analytics), video processing and some ML acceleration.
- Our modular approach is ideal for data scientists exploring various combinations of accelerator cores (you can plug bitstreams together in tens of milliseconds)
- Load accelerators as needed (e.g., dedicated de-compression for different data types)
- Take advantage of natural configuration points (again: sorting is a blocking operation)
- Our framework could work as a SQL / data analytics domain compiler (+ software stack)

Contributors

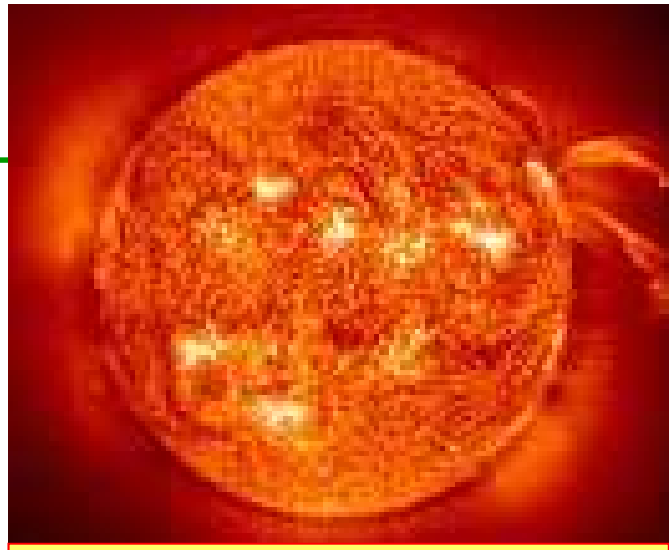
Malte Vesper malte.vesper@manchester.ac.uk (SSD stream processing infrastructure & applications)
Kaspar Matas kaspar.matas@manchester.ac.uk (Middleware for dynamic stream processing)
Christian Beckhoff (GoAhead support)
Khoa Pham khoa.pham@manchester.ac.uk (HLS support for PR and runtime management)
Anuj Vaishnav anuj.vaishnav@manchester.ac.uk (Resource-elastic FPGA virtualization & cloud infrastructure)
Kristiyan Manev kristiyan.manev@postgrad.manchester.ac.uk (Resource-elastic stream processing)
Babis Kritikakis babis_k4@hotmail.com (Dynamic Dataflow on Maxeler)
Tuan La tuan.la@manchester.ac.uk (FPGA hardware security)
Joe Powell joe-powell@manchester.ac.uk (FPGA hardware security)
Dirk Koch dirk.koch@ziti.uni-heidelberg.de

Find all our projects on git: https://github.com/orgs/FPGA-Research-Manchester/

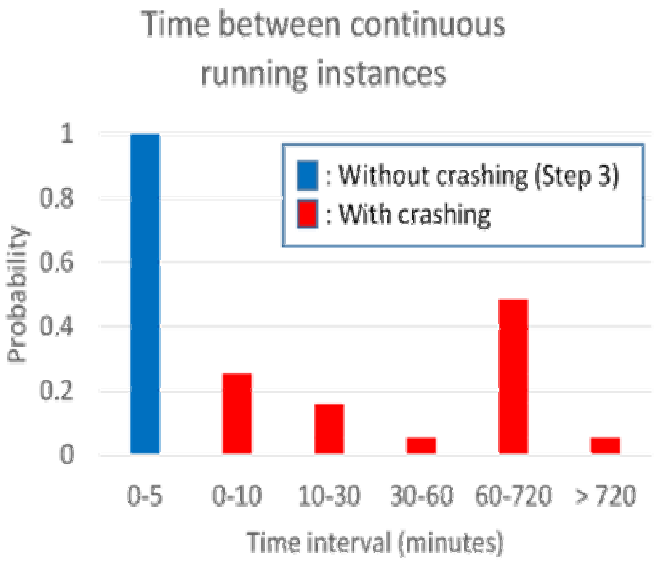
- **GoAhead** **https://github.com/FPGA-Research-Manchester/GoAhead**
- **FOS** **https://github.com/FPGA-Research-Manchester/fos**
- **Dynamic Streams** **https://github.com/FPGA-Research-Manchester/OrkhestraFPGAStream**
- **FPGA Virus Scanning** **https://github.com/FPGA-Research-Manchester/FPGAVirusScanner**

One slide on our FPGA security work

- Sidechannel attacks: no real threat
  (but fault injection, possibly Trojans)
- Power-Hammering Potential (Alveo U200)
  - Oscilators and glitch amplifiers > 2 KW
  - Wires > 5 KW
  - Flops > 2 KW
  - + BRAMs (write collisions), DSPs, Gbit transceivers...



- The sun emits ~63 MW / m^2
- or ~6.3 KW / cm^2



- DoS attack on AWS F1
  (we crashed >100 F1s, see TCHES 2021)
- Bitstream virus scanning
  →reject malicious designs (TRETS 2020)
- FPGA TEEs (FCCM 2021)
- FPGA health monitoring