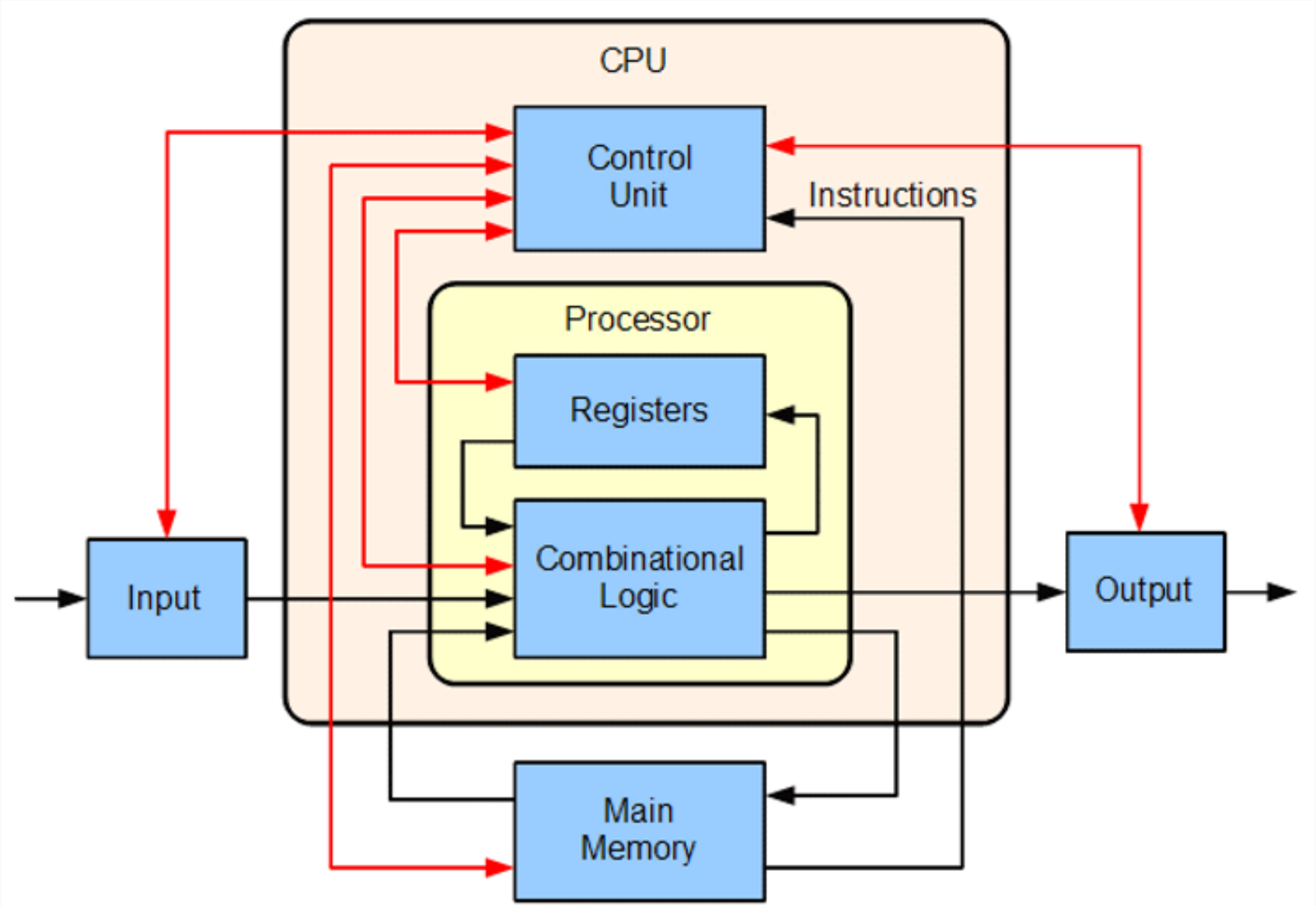
The ExCALIBUR Hardware and Enabling Software (H&ES) Programme

PI: Adrian Jackson (EPCC)

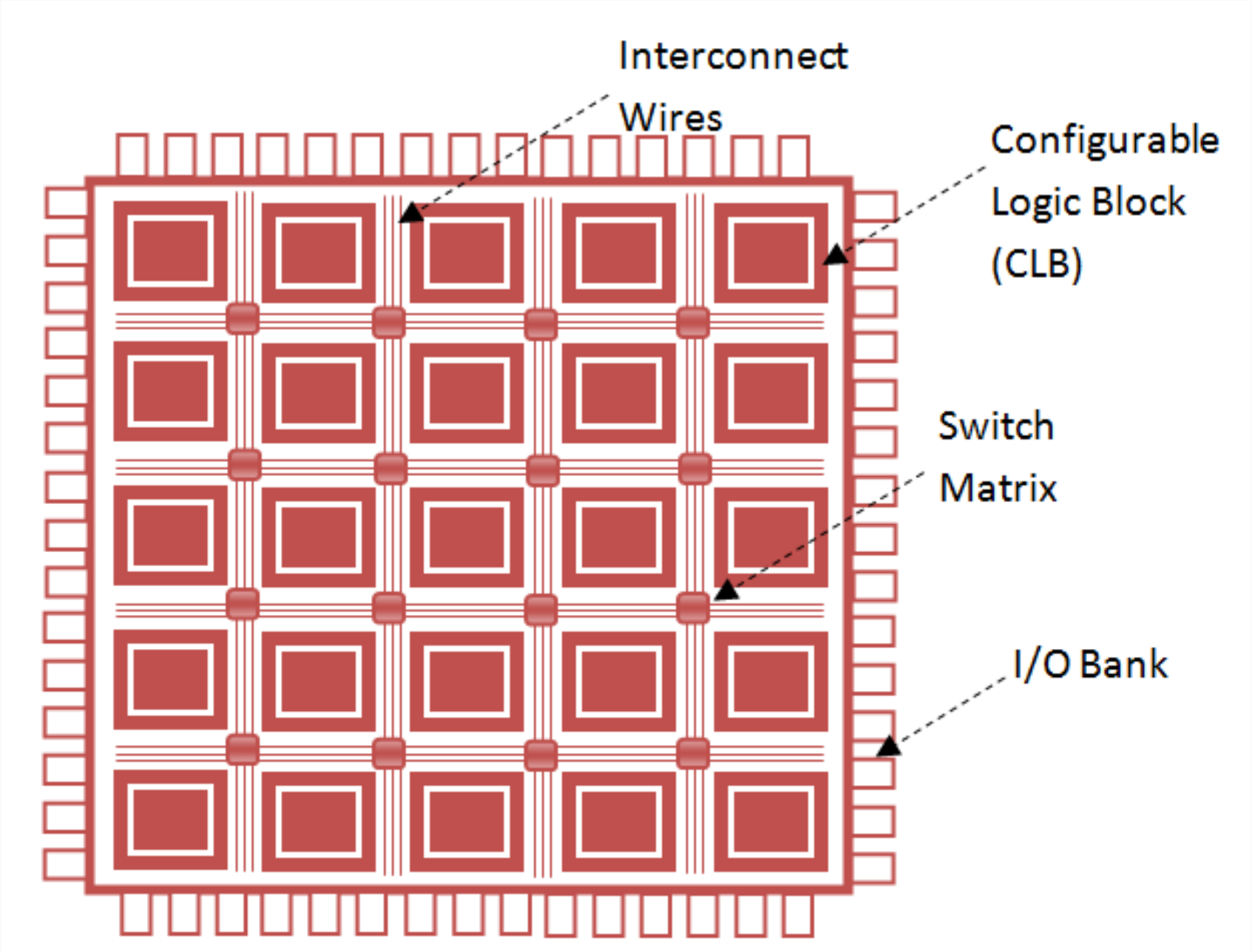
Co-I: Nick Brown (EPCC), Serge Guillas (UCL), Jonathan Cooper (UCL), Suhaib Fahmy (Warwick)

What are FPGAs and why are they useful?

CPU/GPU Architecture



FPGA Architecture

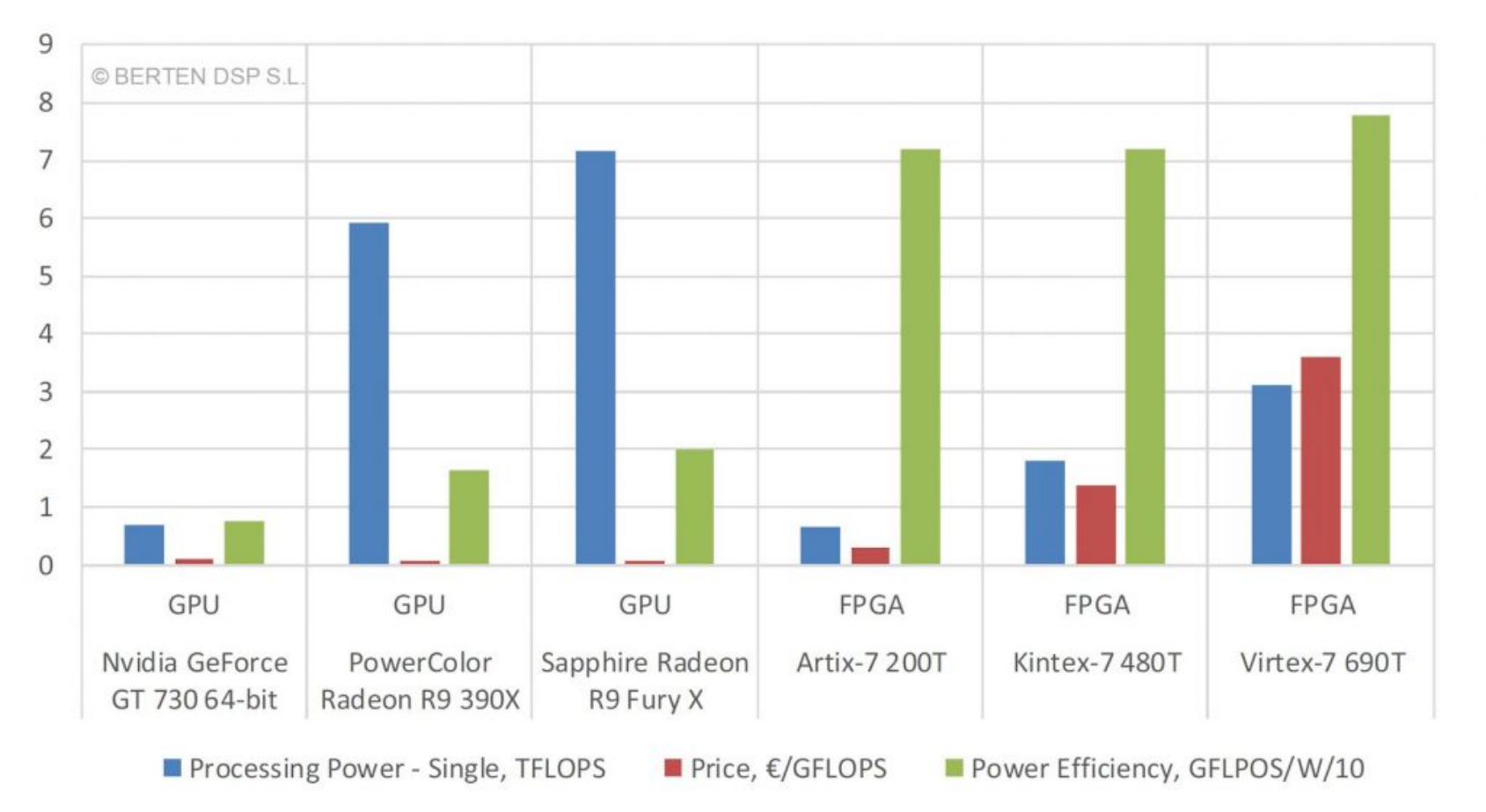


- Core idea of FPGA acceleration is **Single instruction, multiple data** (similar to GPUs)

- Core idea of FPGA acceleration is **Single instruction, multiple data** (similar to GPUs)
- Key difference is **no instruction set** so no time/energy wasted on fetch/decode/execute cycles

- Core idea of FPGA acceleration is **Single instruction, multiple data** (similar to GPUs)
- Key difference is **no instruction set** so no time/energy wasted on fetch/decode/execute cycles
- GPUs offer massive parallelism through many cores (e.g. many factories)

- Core idea of FPGA acceleration is **Single instruction, multiple data** (similar to GPUs)
- Key difference is **no instruction set** so no time/energy wasted on fetch/decode/execute cycles
- GPUs offer massive parallelism through many cores (e.g. many factories)
- FPGAs offer parallelism through **pipelining** (e.g. single production line)



[1]

[1] GPU vs FPGA Performance Comparison - BERTEN. (2020, December 06). Retrieved from https://www.bertendsp.com/gpu-vs-fpga-performance-comparison

FPGA development cycle

- Most performant/difficult is direct circuit design (e.g. Verilog)

FPGA development cycle

- Most performant/difficult is direct circuit design (e.g. Verilog)
- Alternative is high level synthesis (HLS)

FPGA development cycle

- Most performant/difficult is direct circuit design (e.g. Verilog)
- Alternative is high level synthesis (HLS)
- Describe the algorithm then let tools design the circuit

FPGA development cycle

- Most performant/difficult is direct circuit design (e.g. Verilog)
- Alternative is high level synthesis (HLS)
- Describe the algorithm then let tools design the circuit
- This is **slow**

FPGA development cycle

- Most performant/difficult is direct circuit design (e.g. Verilog)
- Alternative is high level synthesis (HLS)
- Describe the algorithm then let tools design the circuit
- This is **slow**
- Fast development requires hardware simulation or CPU portable code

FPGA development cycle

- Most performant/difficult is direct circuit design (e.g. Verilog)
- Alternative is high level synthesis (HLS)
- Describe the algorithm then let tools design the circuit
- This is **slow**
- Fast development requires hardware simulation or CPU portable code
- FPGA optimisation can be unintuitive; requires good optimisation tools

Current state of developer tools

Tool	Vendor Support	Ease of use
Vitis HLS	Xilinx	low-level; includes libraries; good but tricky tools
OpenCL	Xilinx/Intel	low-level; potentially portable to CPU/GPU
SYCL	Intel	better syntax; likely future direction for Intel
DaCe Python	Xilinx/Intel	Higher level; portable; immature; harder to debug + optimise

H&ES Project

- Monte-Carlo Markov Chain application

H&ES Project

- Monte-Carlo Markov Chain application
- Iterative solver library

H&ES Project

- Monte-Carlo Markov Chain application
- Iterative solver library
- Development of training materials & tutorials

H&ES Project

- Monte-Carlo Markov Chain application
- Iterative solver library
- Development of training materials & tutorials
- Potential collaboration with SysGenX

Monte-Carlo Markov Chain (MCMC) generation

- Used to infer likely parameters + detailed error estimates from real/simulated data

Monte-Carlo Markov Chain (MCMC) generation

- Used to infer likely parameters + detailed error estimates from real/simulated data
- Calculates an MCMC using Metropolis-Hastings algorithm

Monte-Carlo Markov Chain (MCMC) generation

- Used to infer likely parameters + detailed error estimates from real/simulated data
- Calculates an MCMC using Metropolis-Hastings algorithm
- Computationally heavy parts are...

Monte-Carlo Markov Chain (MCMC) generation

- Used to infer likely parameters + detailed error estimates from real/simulated data
- Calculates an MCMC using Metropolis-Hastings algorithm
- Computationally heavy parts are...
  - Covariance matrix generation

Monte-Carlo Markov Chain (MCMC) generation

- Used to infer likely parameters + detailed error estimates from real/simulated data
- Calculates an MCMC using Metropolis-Hastings algorithm
- Computationally heavy parts are...
  - Covariance matrix generation
  - Cholesky decompositions

Monte-Carlo Markov Chain (MCMC) generation

- Used to infer likely parameters + detailed error estimates from real/simulated data
- Calculates an MCMC using Metropolis-Hastings algorithm
- Computationally heavy parts are...
  - Covariance matrix generation
  - Cholesky decompositions
- Current CPU codes limited to $O(100)$ samples

Monte-Carlo Markov Chain (MCMC) generation

- Used to infer likely parameters + detailed error estimates from real/simulated data
- Calculates an MCMC using Metropolis-Hastings algorithm
- Computationally heavy parts are...
  - Covariance matrix generation
  - Cholesky decompositions
- Current CPU codes limited to $O(100)$ samples
- Used development to better understand Vitis HLS tooling

Iterative Solvers library

- Identified as missing niche in FPGA space

Iterative Solvers library

- Identified as missing niche in FPGA space
- Plan is to...

Iterative Solvers library

- Identified as missing niche in FPGA space
- Plan is to...
  - Extend and benchmark existing solvers

Iterative Solvers library

- Identified as missing niche in FPGA space
- Plan is to...
  - Extend and benchmark existing solvers
  - Adapt solvers for Intel FPGAs

Iterative Solvers library

- Identified as missing niche in FPGA space
- Plan is to...
  - Extend and benchmark existing solvers
  - Adapt solvers for Intel FPGAs
  - Implement vendor-agnostic front-end

Iterative Solvers library

- Identified as missing niche in FPGA space
- Plan is to...
  - Extend and benchmark existing solvers
  - Adapt solvers for Intel FPGAs
  - Implement vendor-agnostic front-end
  - Focus on usability

Iterative Solvers library

- Identified as missing niche in FPGA space
- Plan is to...
  - Extend and benchmark existing solvers
  - Adapt solvers for Intel FPGAs
  - Implement vendor-agnostic front-end
  - Focus on usability
  - Integrate with existing codes/projects (WP4 of SysGenX?)