A molecular dynamics code on CPU, GPU and FPGA

Matthieu Haefele, CNRS/UPPA

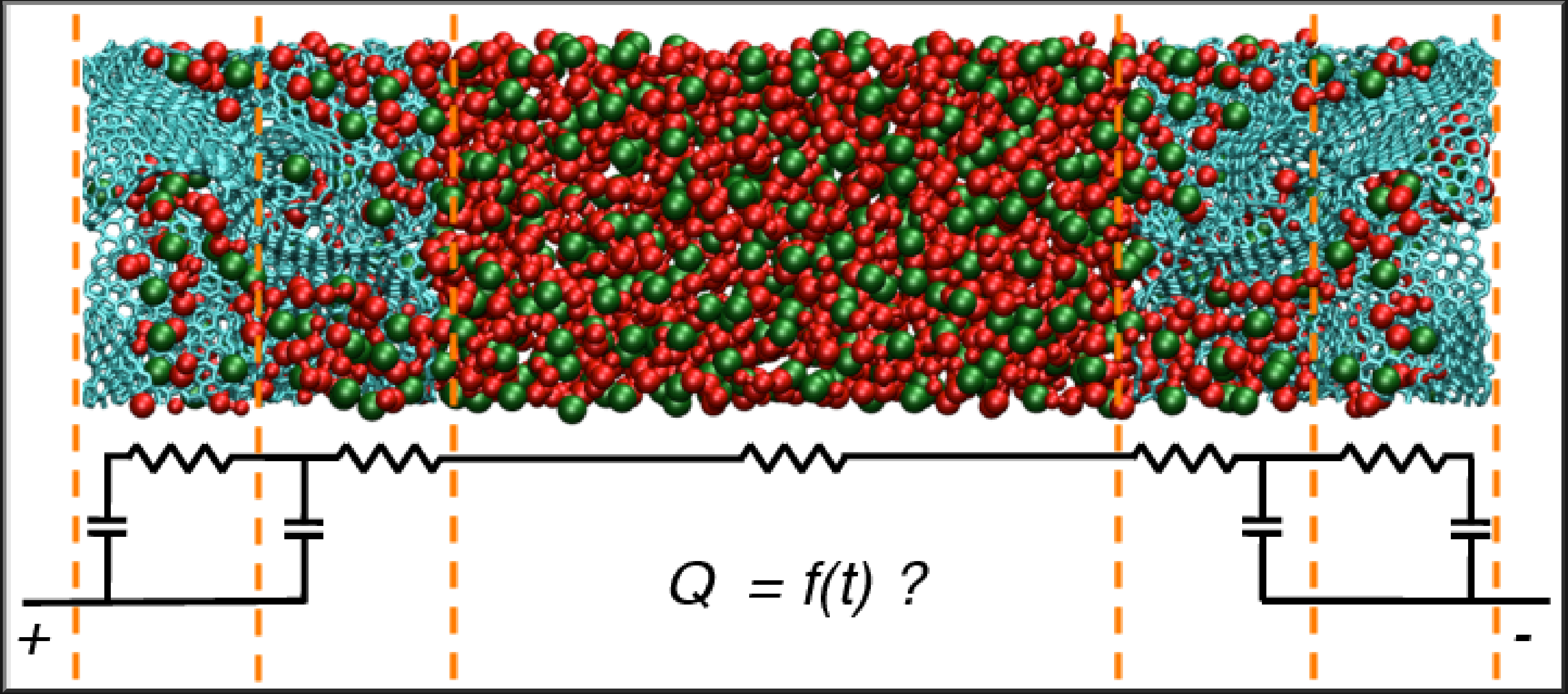
Pau, Novembre 2019

Outline

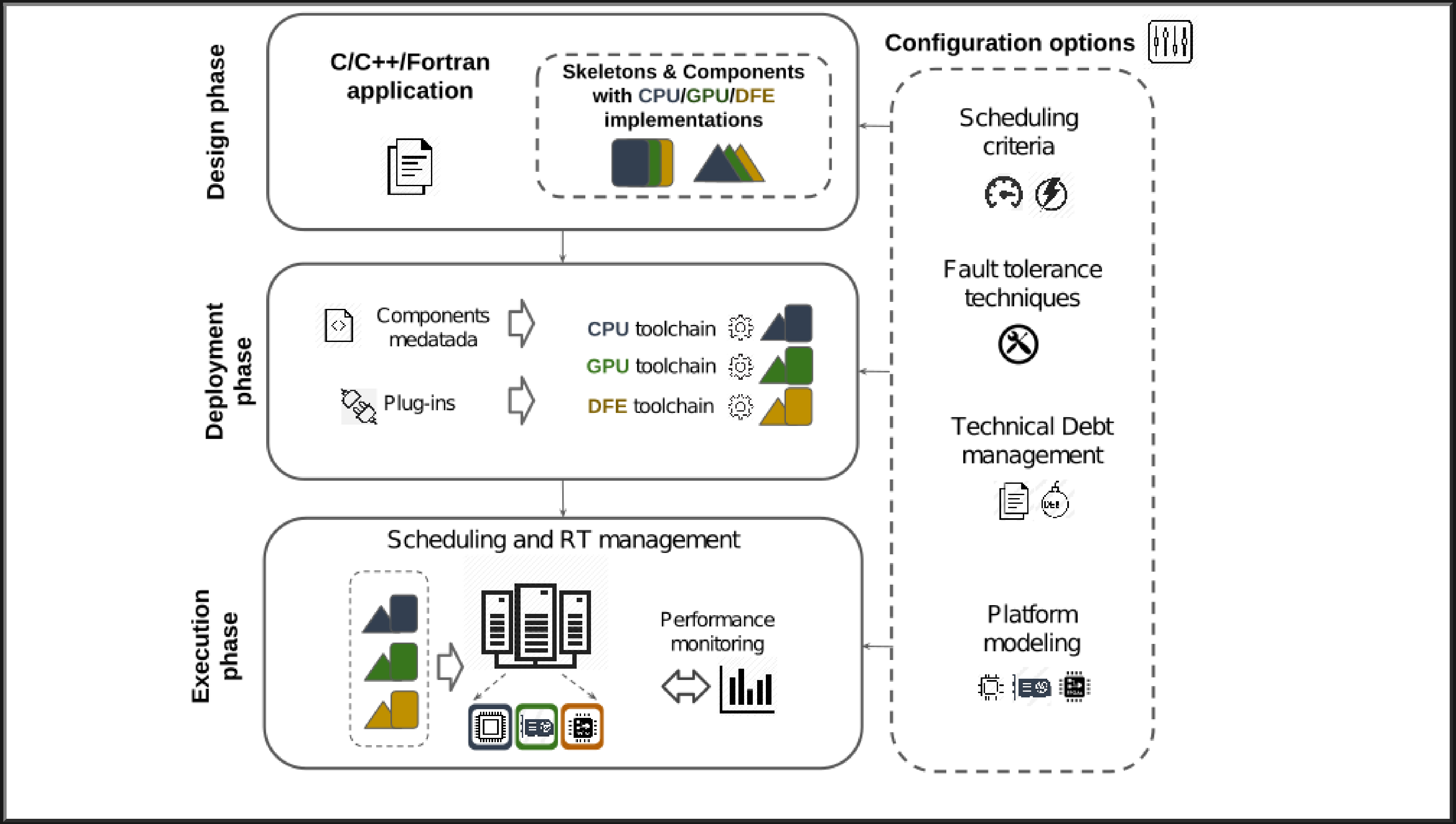
- Introduction
- Porting on GPUs
- Comparison CPU vs GPU in time and energy
- Current effort on FPGA porting

Supercapacitors simulation

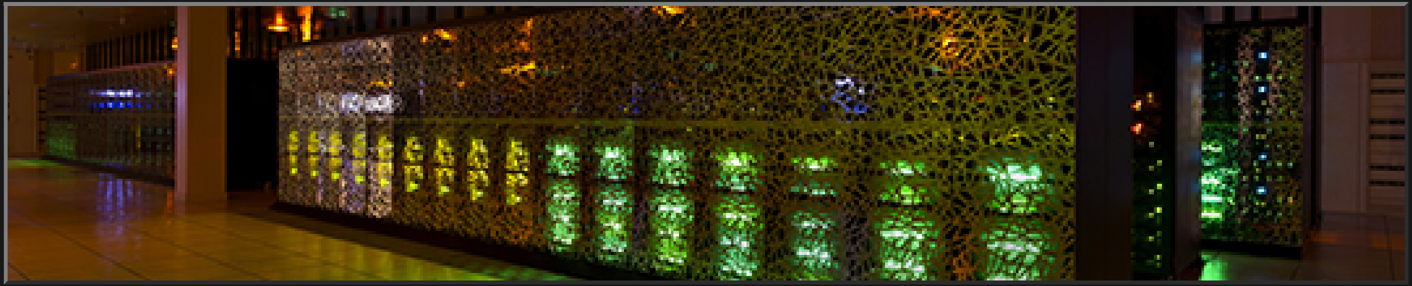
- Carbon electrodes (Blue), Ionic liquid (red/green)
- Study of the liquid/electrode interface
- Molecular dynamics with accurate electrostatic
- Constant potential electrodes
- Code Metalwalls, maintained by M. Salanne (Sorbonne University)



EXA2PRO H2020 FET-HPC project



Performance improvement on CPU



- Initially F90, pure MPI code (20K lines)
- x15 in performance
- 28 MCPUh allocated last four years => 60 MCPUh saved
- Enables science that could not be reached
- Hybrid MPI+OpenMP and MPI+OpenACC now available
- Single maintainable code base

GPU porting with OpenAcc

- Simple data structures in Metalwalls => ok for GPU
- First `!$acc kernels` done alone thanks to OpenAcc tutorials
- GPU hackathon @IDRIS solved the last two key issues
- 3-4 months of work in total
- Porting of 7K lines of intensive computations with ~800 directives
- Working Multi-GPU implementation on OpenPower (ouessant)

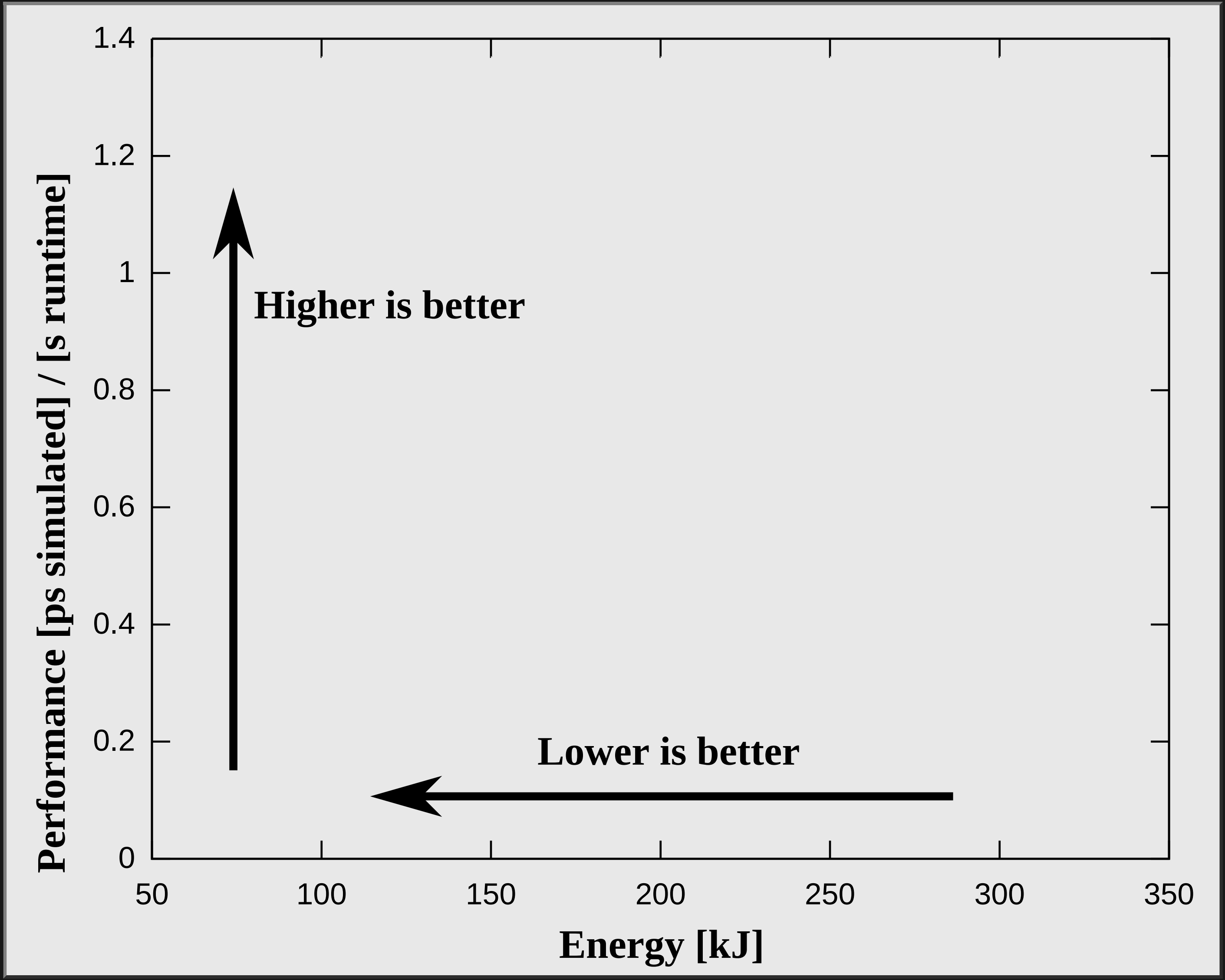
Step 1: getting the correct answer

- Kernel by kernel porting
- Unified memory takes care of data transfer
- Avoid computations on the CPU to avoid CPU/GPU data transfer
- Objective: to have a full time step running on the GPU

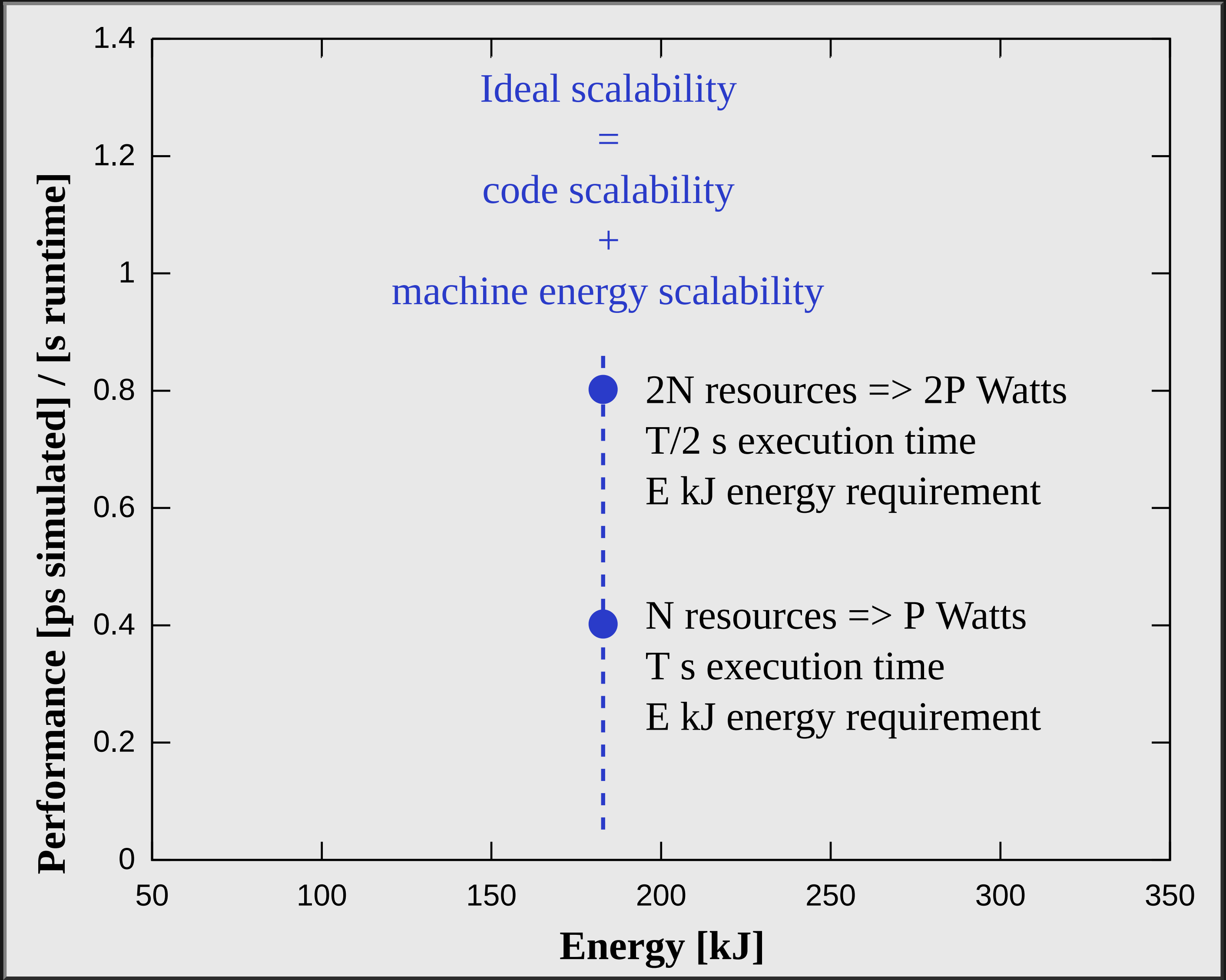
Step 2: getting performance

- Running the NVidia profiler
- Replacing `acc kernels` with `acc parallel loop`
- Adding `loop gang` and `loop vector` => two levels of parallelism
- Adding `collapse` => larger amount of work to distribute
- Switching off the unified memory with explicit data transfers

Architecture comparison



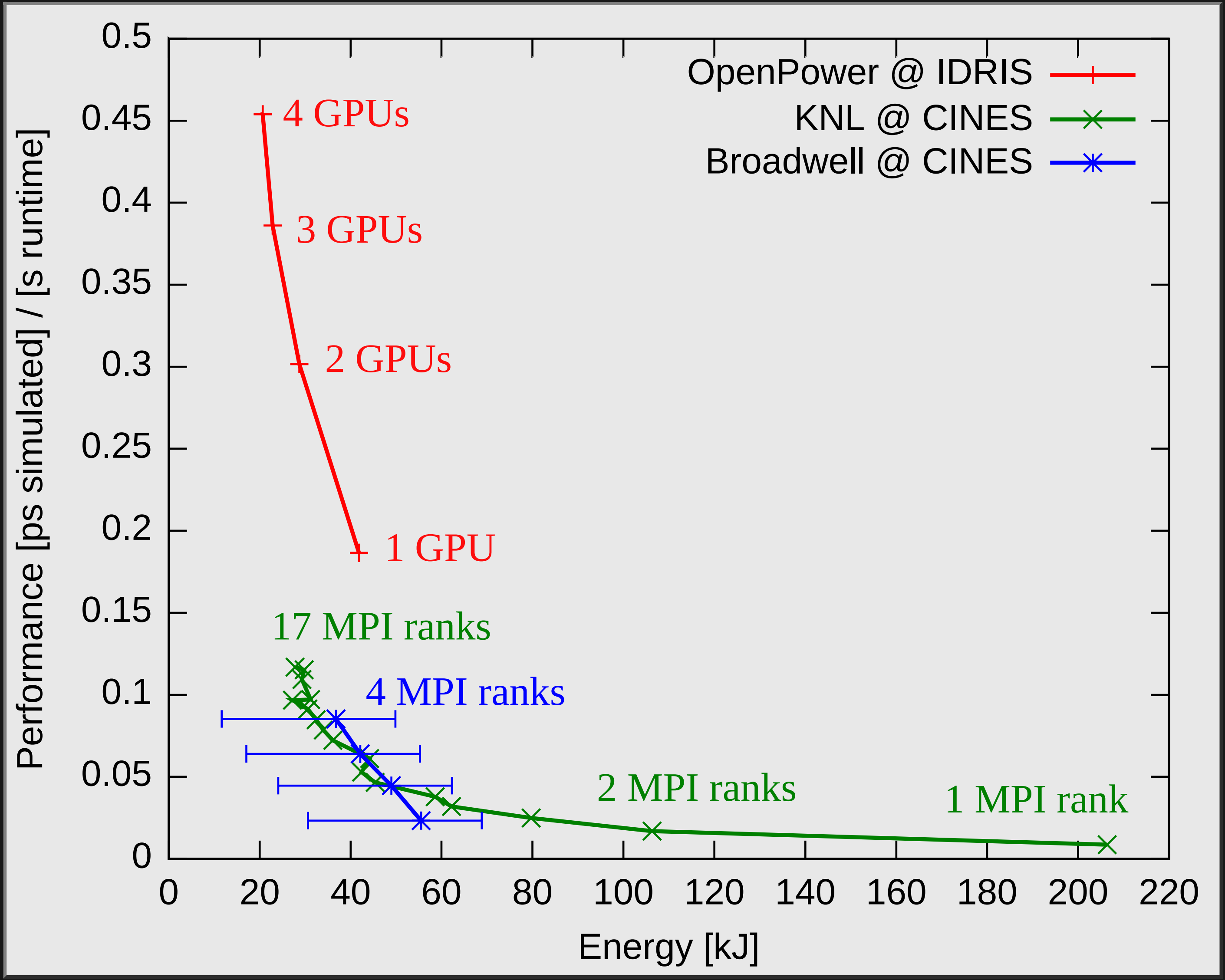
Architecture comparison



Computing systems used

- PRACE PCP system **KNL @ CINES**
  - Intel Xeon Phi, Knights Landing 7250@1.40GHz
  - 17 MPI ranks / node, 8 threads / rank (SMT2)
- GENCI system **Broadwell @ CINES**
  - Intel Xeon Broadwell (E5-2690 V4@2.6GHz)
  - 4 MPI ranks / node, 14 threads / rank (SMT2)
- GENCI system **OpenPower @ IDRIS**
  - Minsky S822LC (Power8+, GPU Pascal, NVLink)
  - 4 MPI ranks / node, 1 GPU per rank

Results intra-node

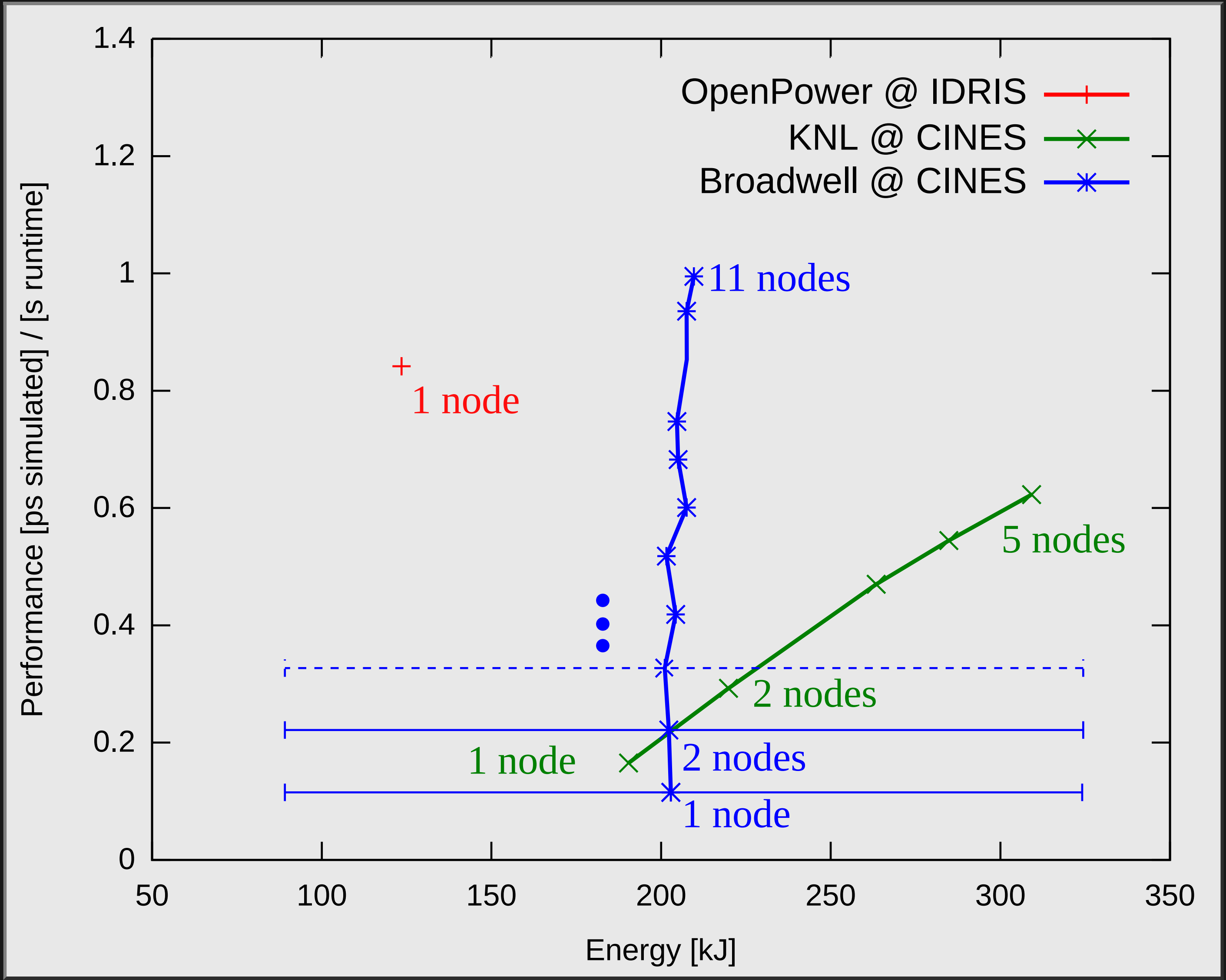


Results intra-node

- Outstanding super linear energy scalability !
- **Wrong**: very poor energy baseline

Dark silicon by just letting resources idle within a node is not an option

Results inter-node



Results inter-node

From **performance** point of view at scalability limit

- BDW is 1, KNL 62%, GPU 84%

From **energy** point of view at scalability limit

- GPU is 1, KNL 2.5

Comment: energy measurement issues on BDW, unfortunately

You said FPGA ?

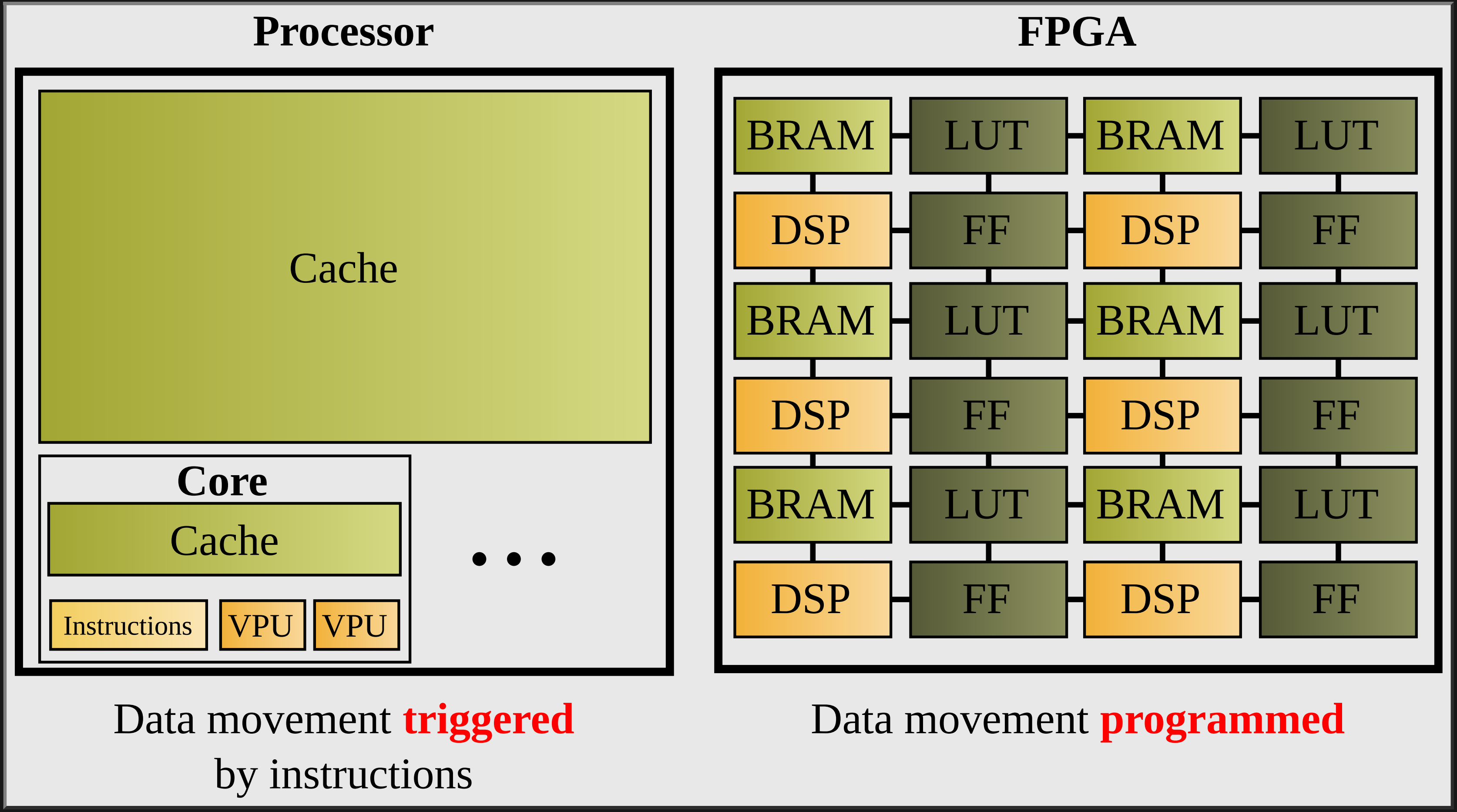
Some acronyms

- FPGA: Field Programmable Gate Array
- LUT: Look Up Table (boolean logic function)
- FF: Flip-Flop (circuit to store one bit of information)
- BRAM: 4KB blocks of RAM
- DSP: Digital Signal Processing (versatile arithmetic unit)

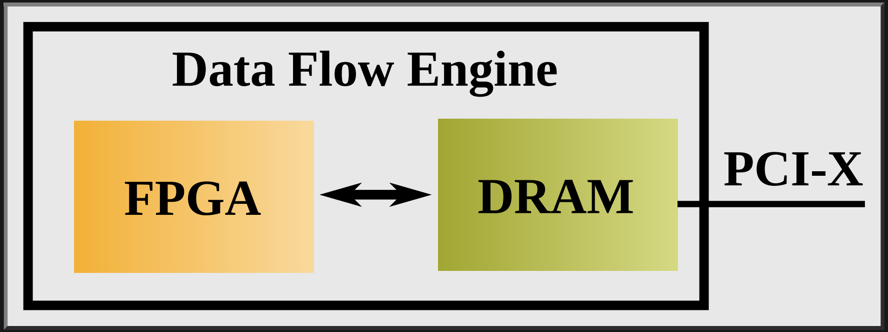
But what is it ?

- Reconfigurable logic
- Algorithm "hard wired" in the silicon
- Computations offloaded as for a GPU accelerator

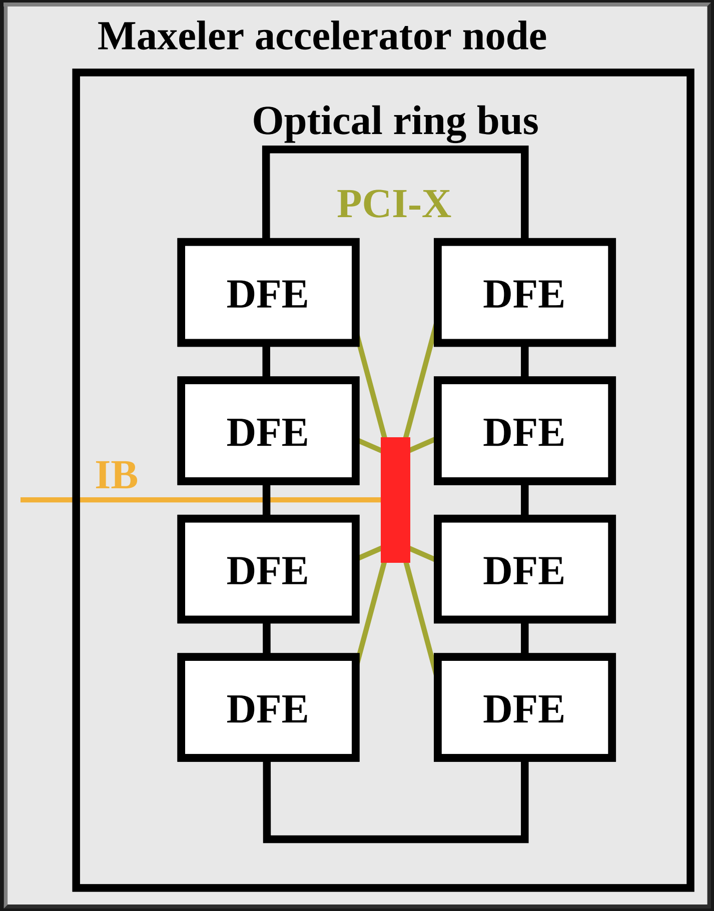
An FPGA is NOT a processor



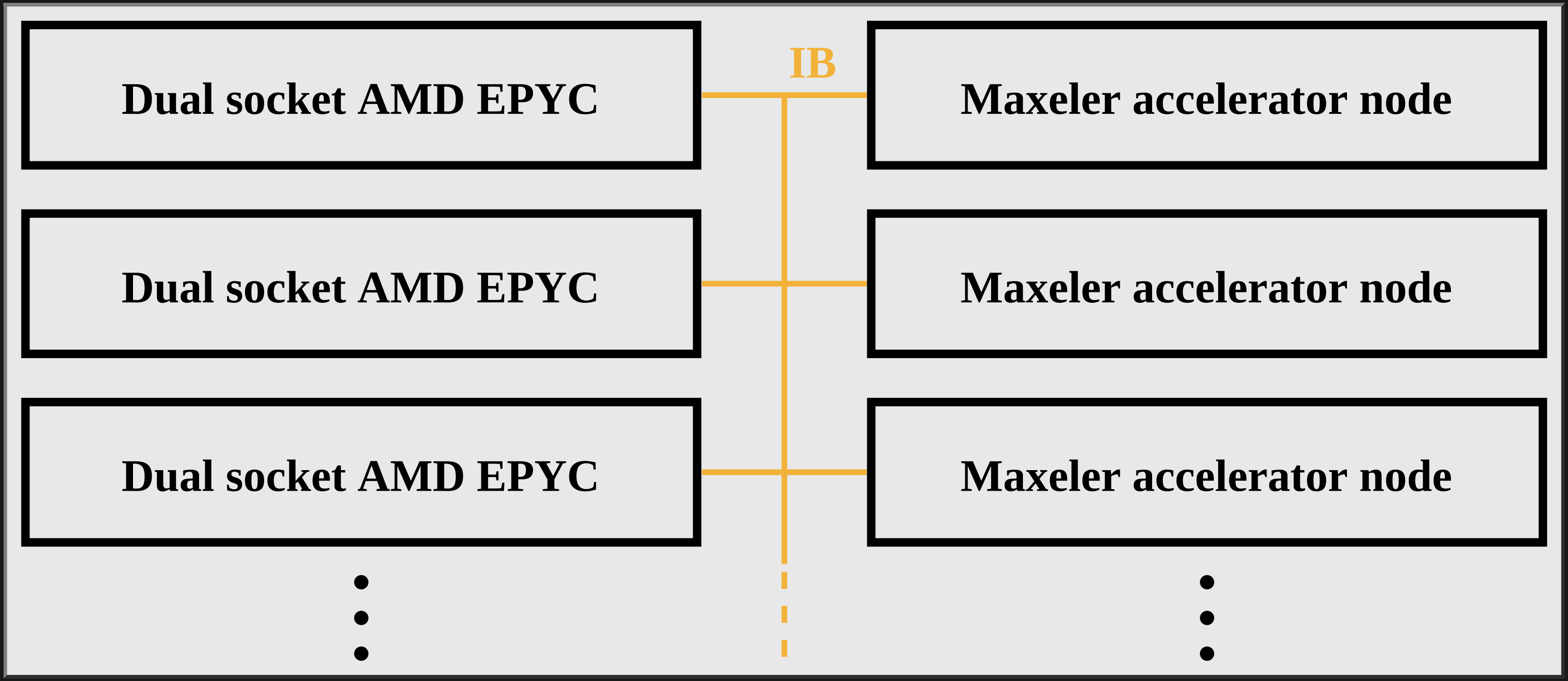
Data Flow Engine



Maxeler accelerator node



Computing system



Architecture comparison

Chip	**2x Intel SkyLake**	**Intel KNL**	**NVidia Pascal**	**Xilinx XCVU9P**
Techno.	14nm	14nm	16nm	16nm
Power	410W	215W	300W	**< 50W**
Freq.	2.7GHz	1.4GHz	1.5GHz	0.1-0.5GHz
cache	2x57MiB	34 MiB	18 MiB	62 MiB
HBM / MCDRAM	0	16GB	16GB	0
DRAM	128-768 GB	384 GB	0	48GB
Peak perf. (DB)	4 TF/s	3 TF/s	5.3 TF/s	**0.5 TF/s**

How to design a circuit for FPGA ?

Available languages

- VHDL, the standard: very low level for electronic people
- XilinX HLS: Pragmas for C
- SycL or Intel OneAPI (C++ framework)
- **Maxeler MaxJ**: Domain Specific Language based on Java

Challenges

- Algorithm reformulation for a streaming implementation
- Reductions are your enemies
- Significant space on silicon can be saved with smaller precision arithmetic

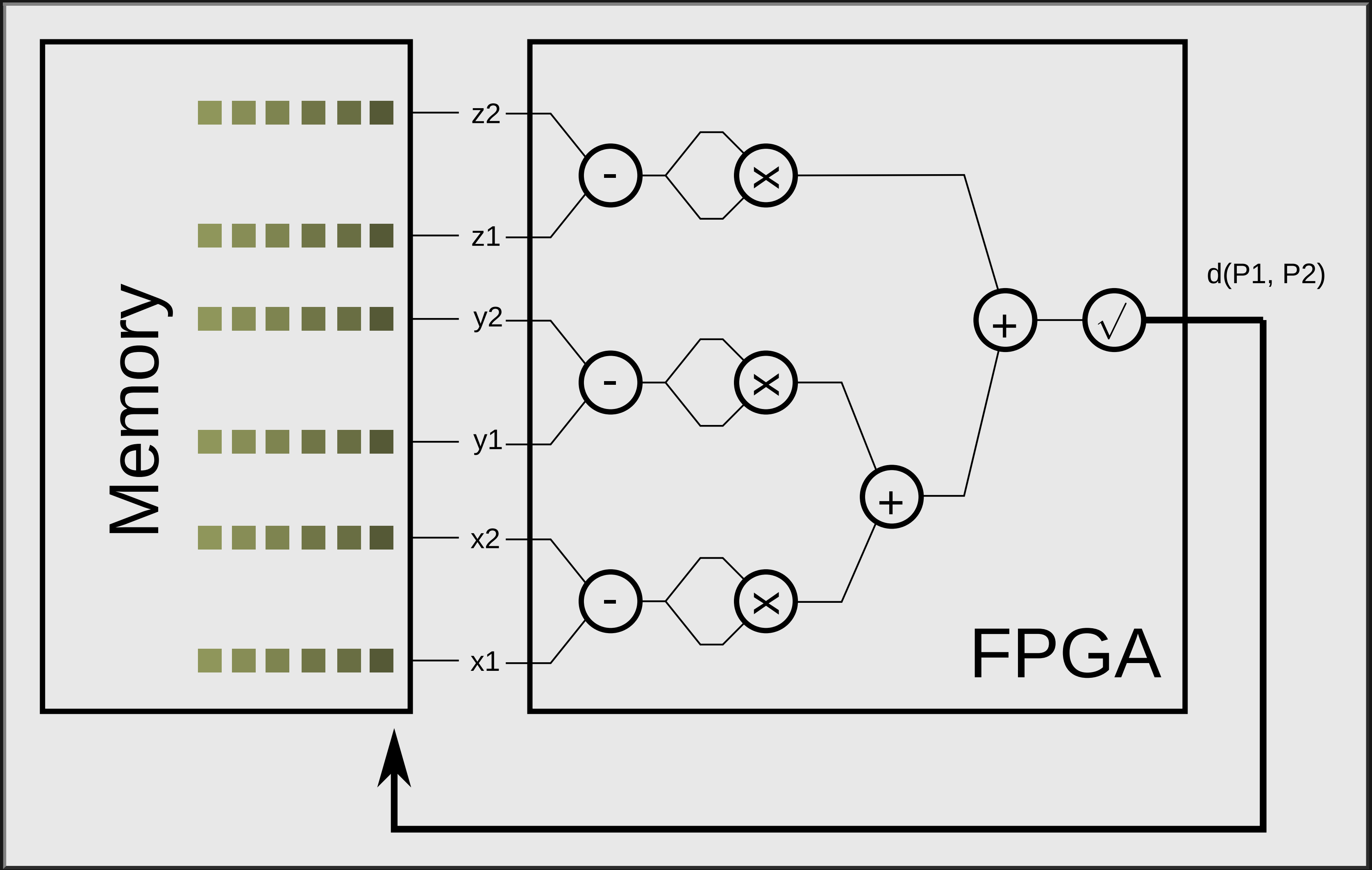
Development workflow

- State of the chip known at each clock tick
- Spreadsheet based performance model reliable (5-10%)
- Design choices performed playing around with LibreOffice

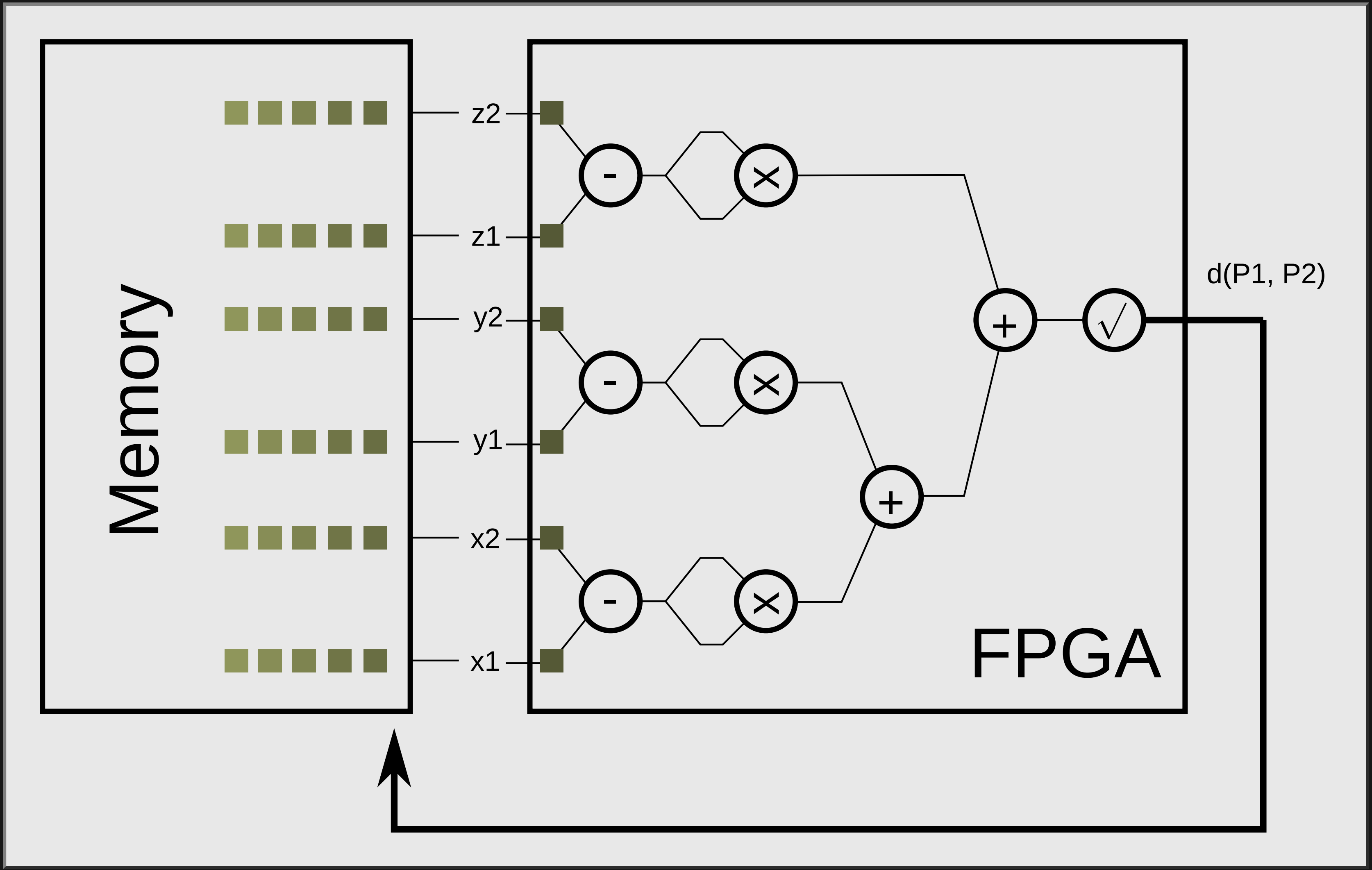
Development workflow

- Kernels written in MaxJ language: embedded DSL based on java
- Eclipse IDE speeds up development and unit tests execution
- Kernels called from F90 or C/C++ with offload mechanism
- **24-48 hours needed for kernels compilation !!**
- Algorithm correctness performed in simulator / emulator

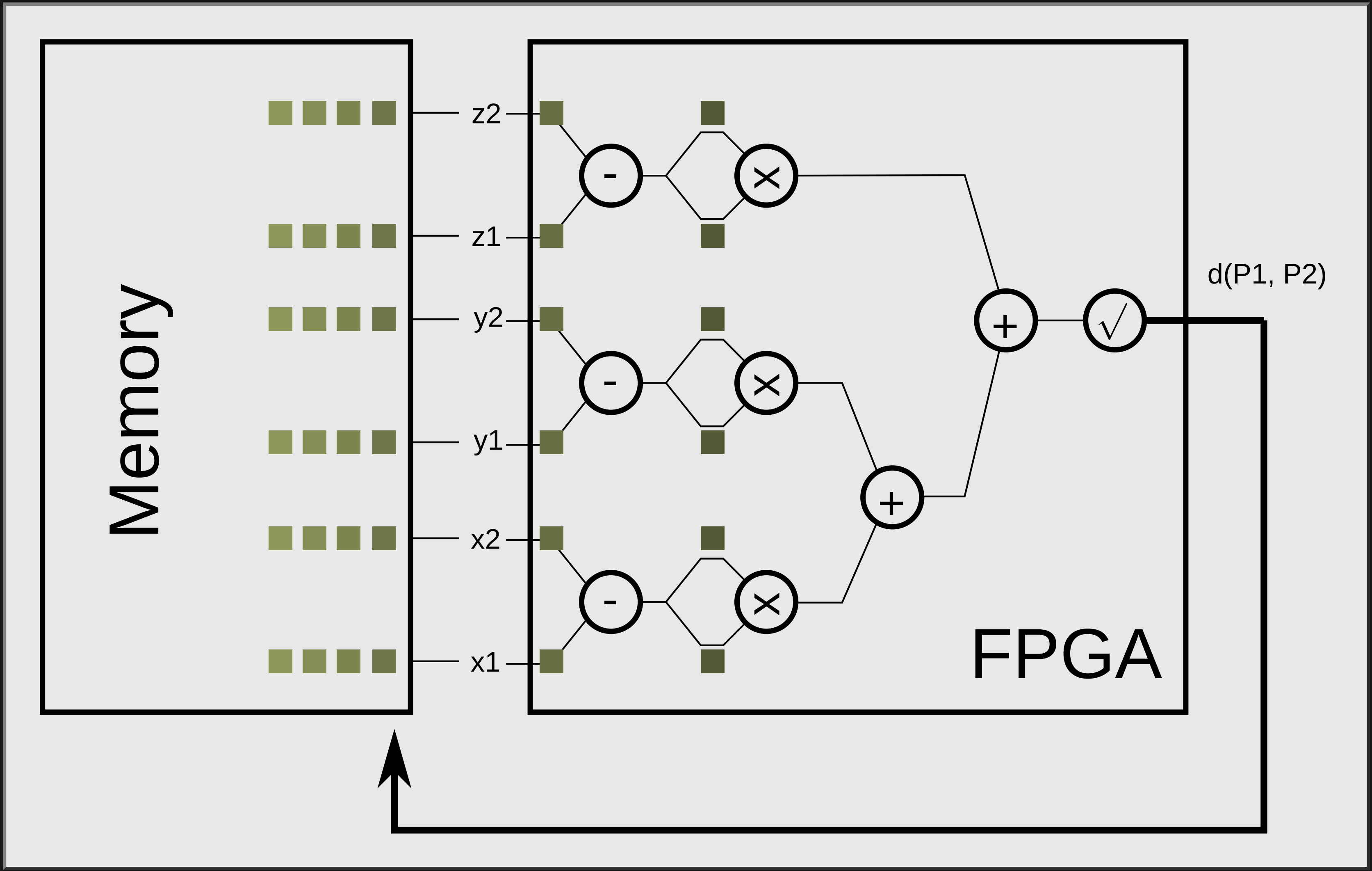
Example: computing a distance



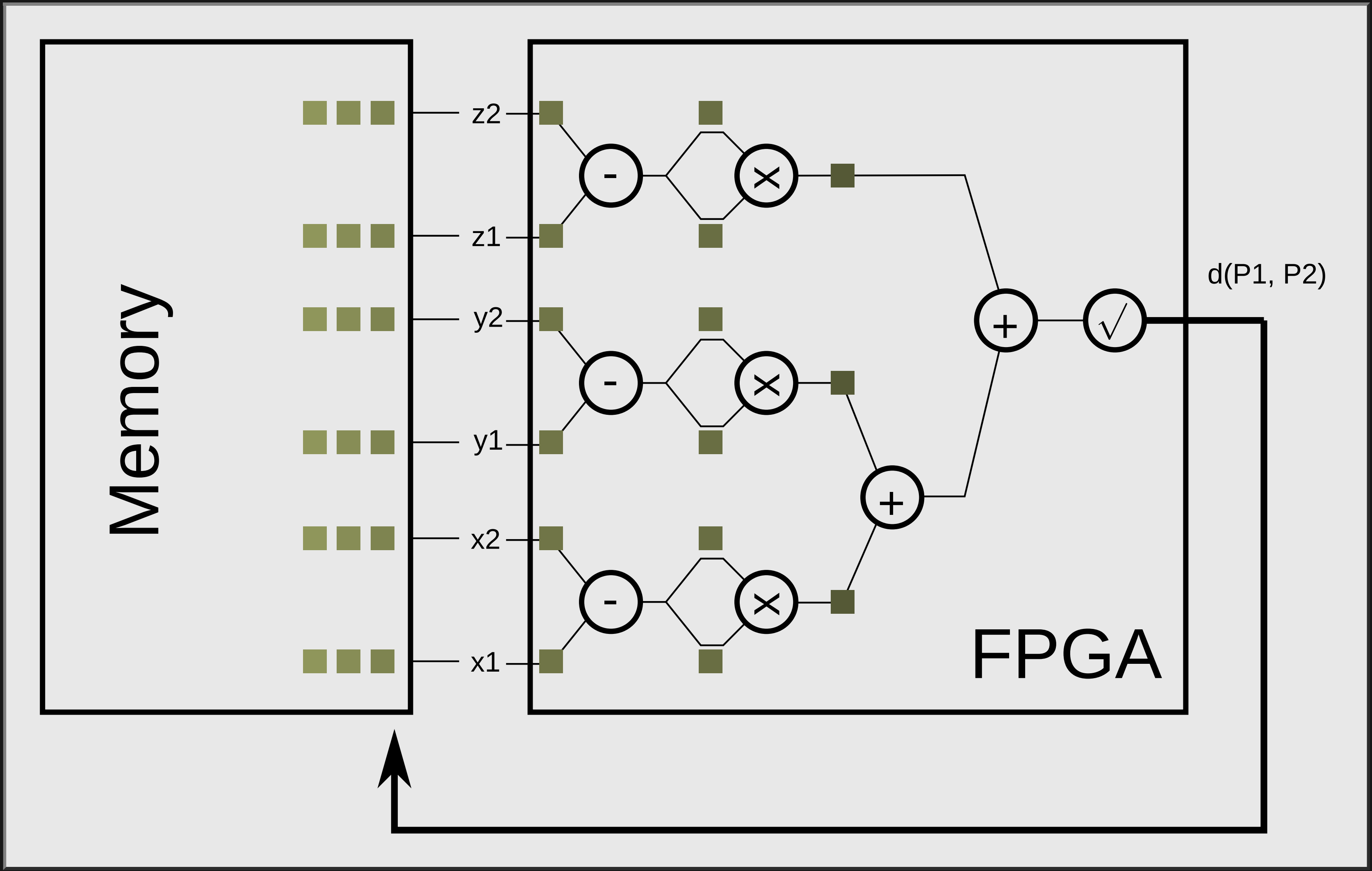
Example: computing a distance



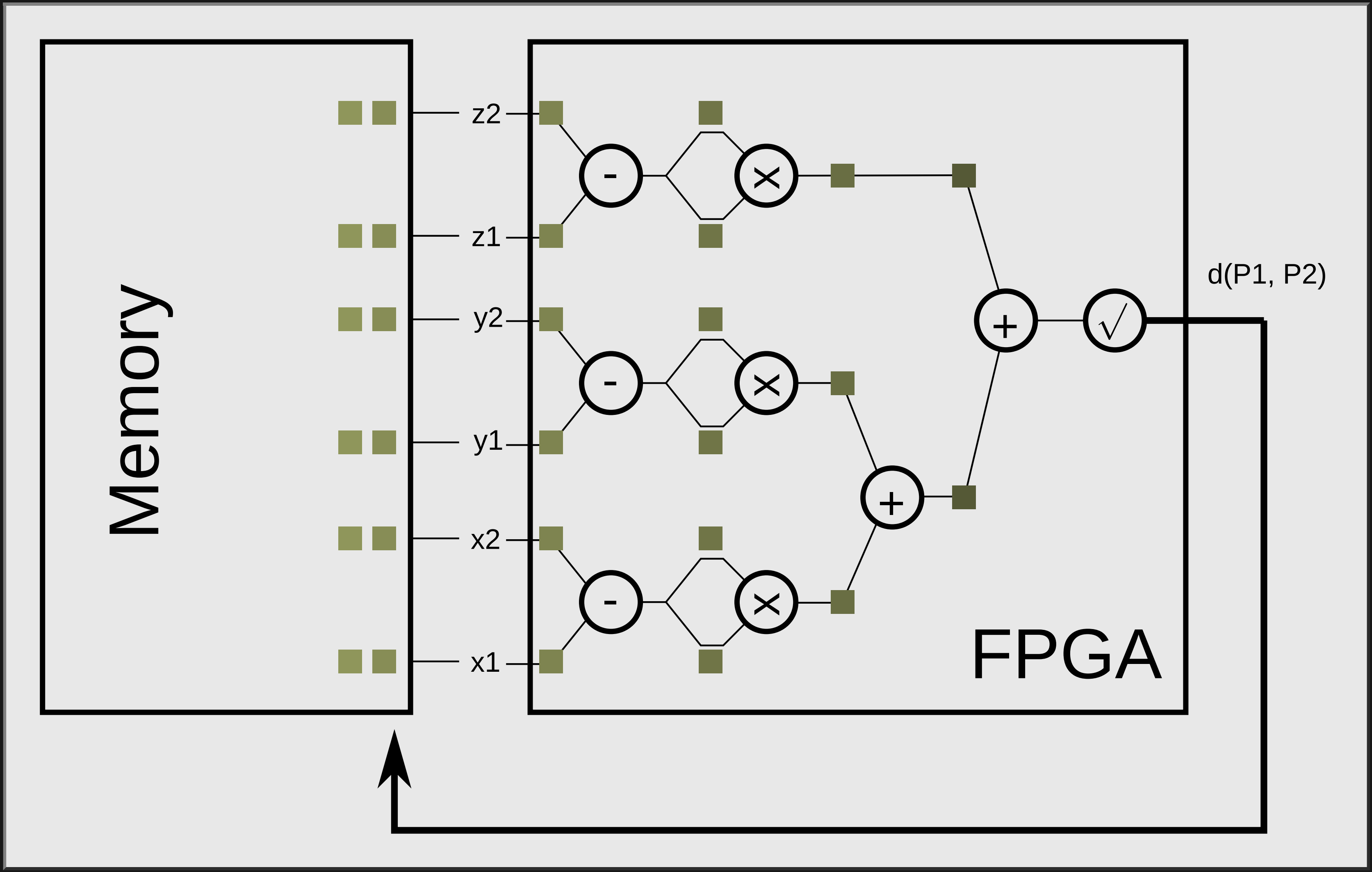
Example: computing a distance



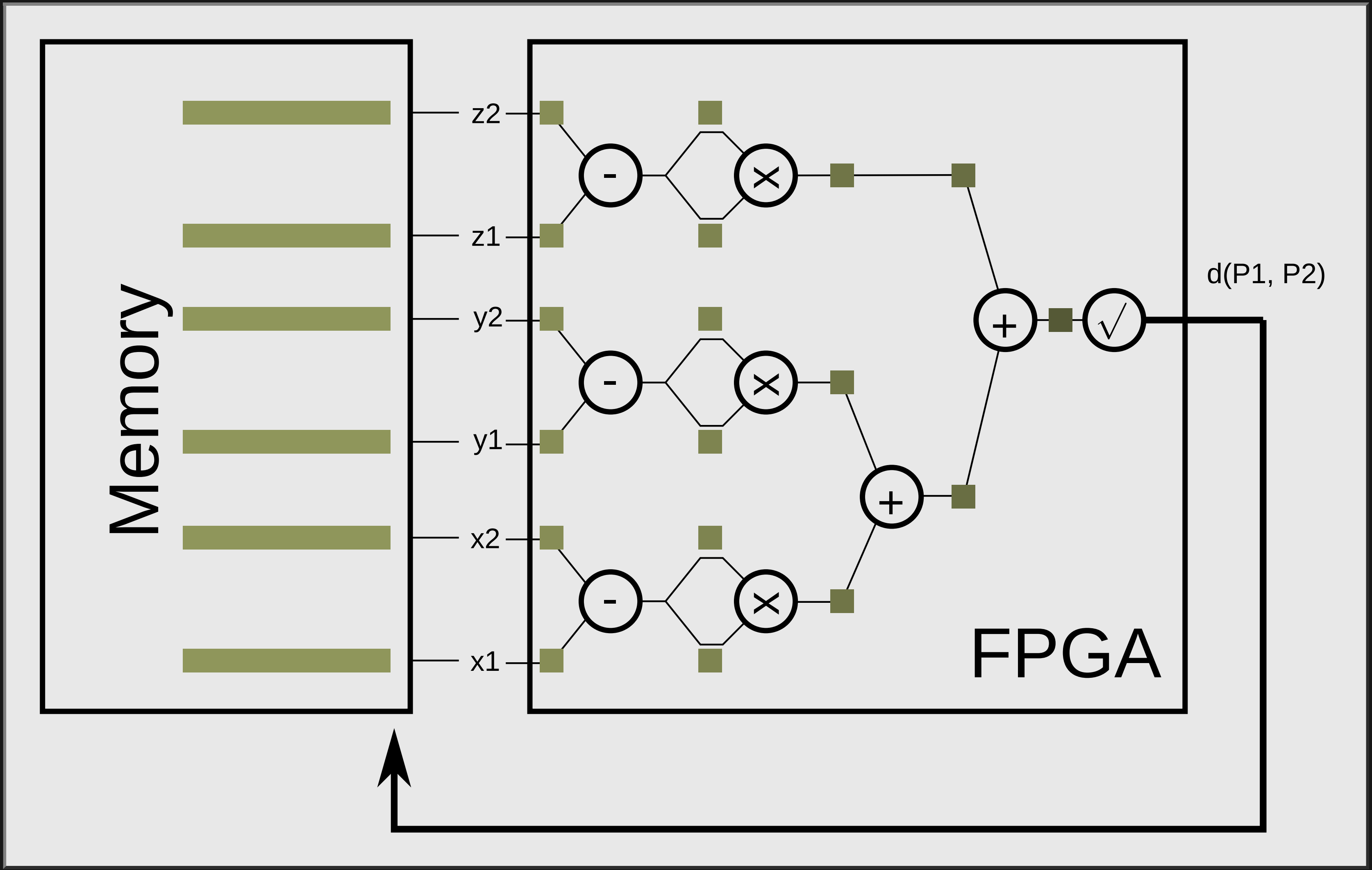
Example: computing a distance



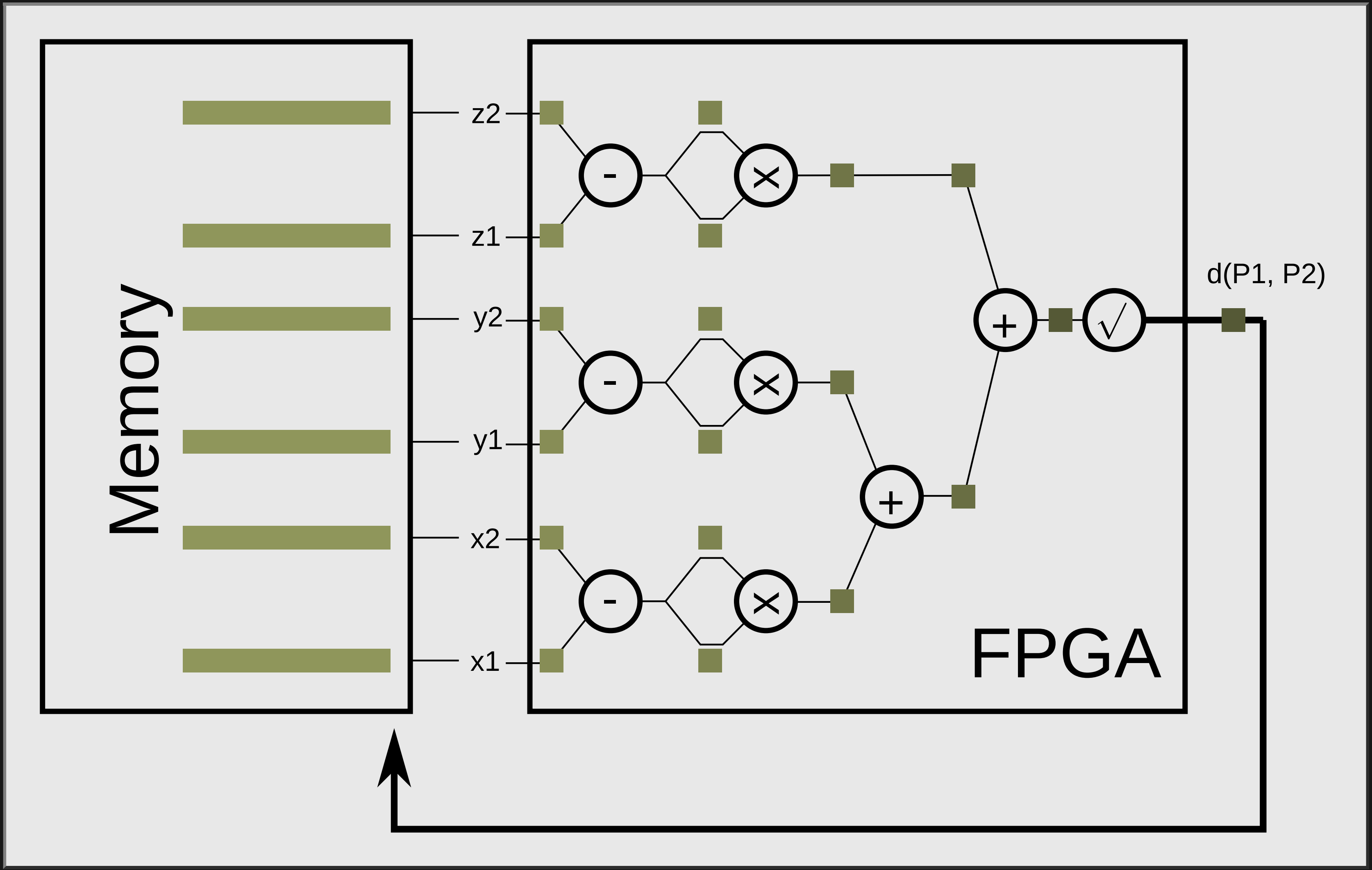
Example: computing a distance



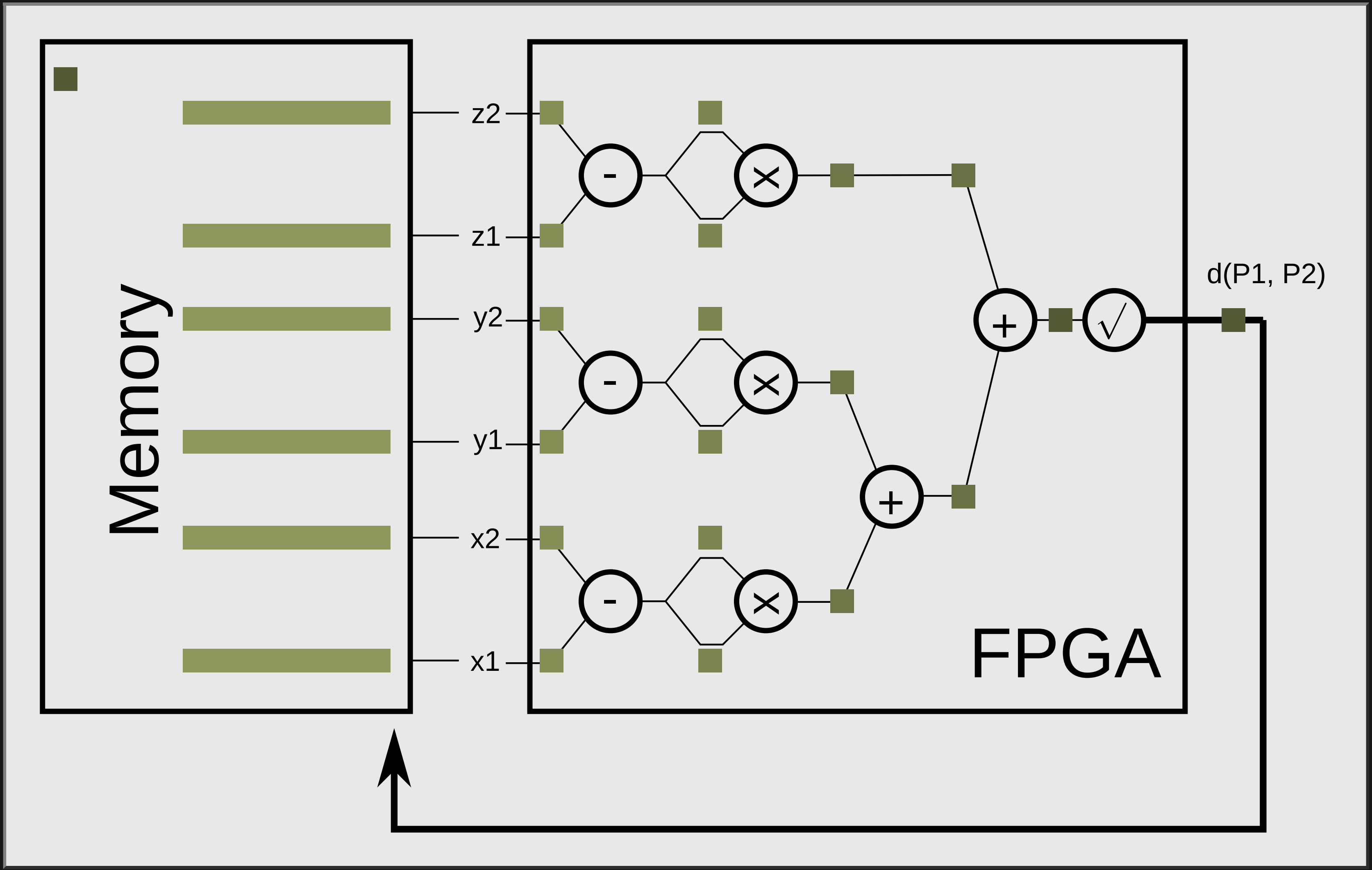
Example: computing a distance



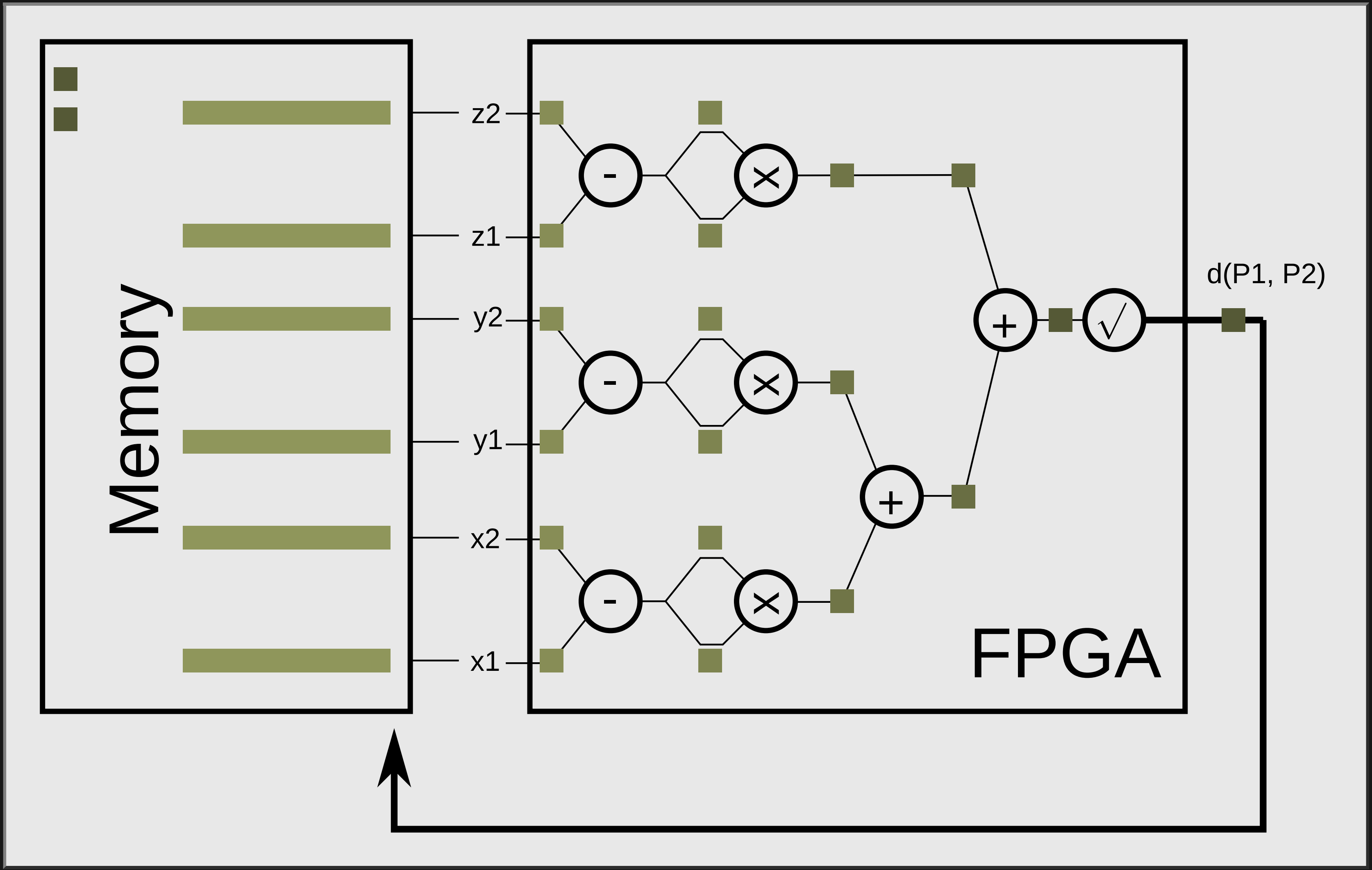
Example: computing a distance



Example: computing a distance



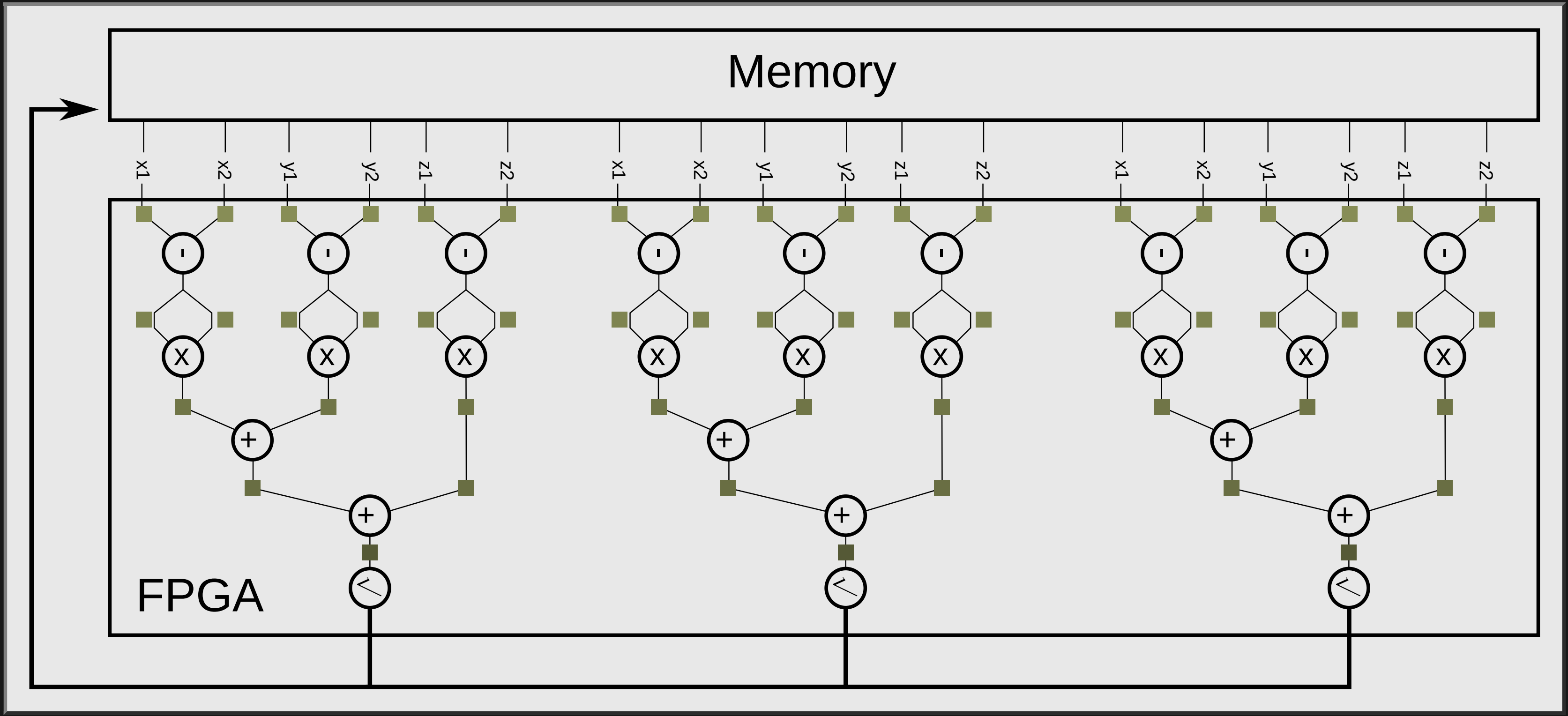
Example: computing a distance



Example: computing a distance

- That was computing in time: building a pipeline
- Now computing in space: replicating this pipeline

Example: computing a distance



Mixed precision implementations

On CPU / GPU

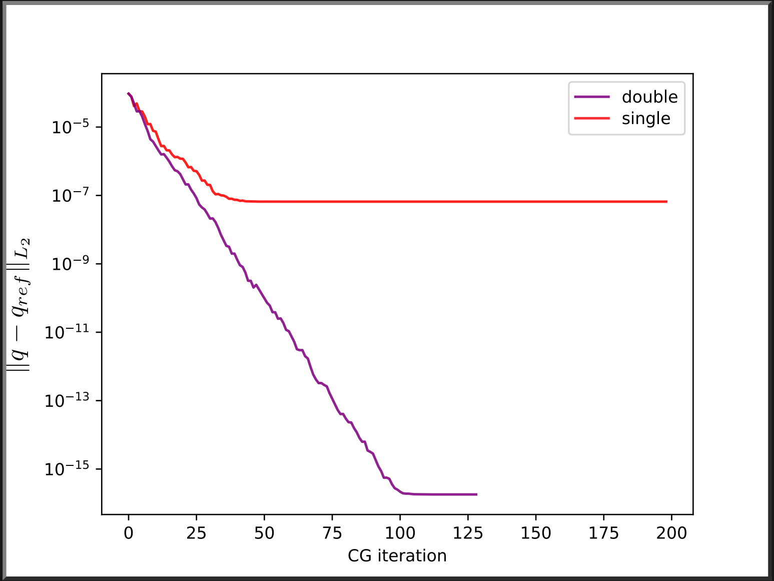
- Double, single and half (only GPU) precision floating point representation
- Factor 2x (resp. 4x) in performance expected when using single (resp. half) instead of double

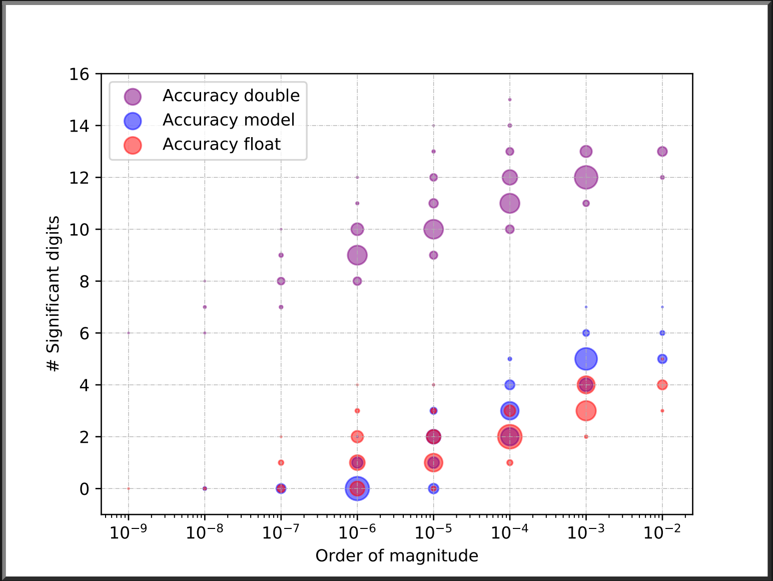
On FPGA

- Any floating point representation available, even fixed point
- Similar performance but requires less resources (2x - 6x between SP and DP)

How could we reduce the accuracy of number representation without damaging the result ?

Numerical accuracy analysis





- Quantifying the accuracy of the results with QP, DP & SP
- Quantifying the accuracy of the model
- Gap between the two: opportunity for DFE performance
- Current status: first kernels running in simulator, first examples run in HW

Conclusion & perspectives

CPU & GPU

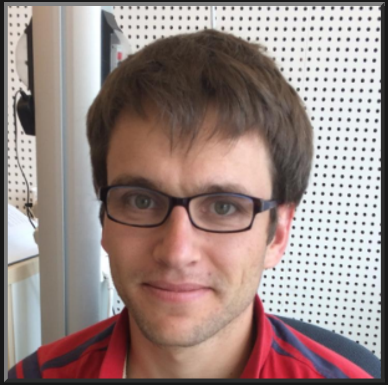
- x15 in performance on CPU
- Code ported on GPU with OpenAcc
- Single maintainable code base
- 4 GPUs 3x more efficient in energy and 25% faster than 5 KNL nodes

FPGA

- Accuracy study done
- FPGA implementation started

Beyond FPGA, the DFE programming model focuses the developer on data movement which is also relevant on CPU & GPU

Acknowledgement


Abel Marin-Laflèche

Vineet Soni

Charles Prouveur