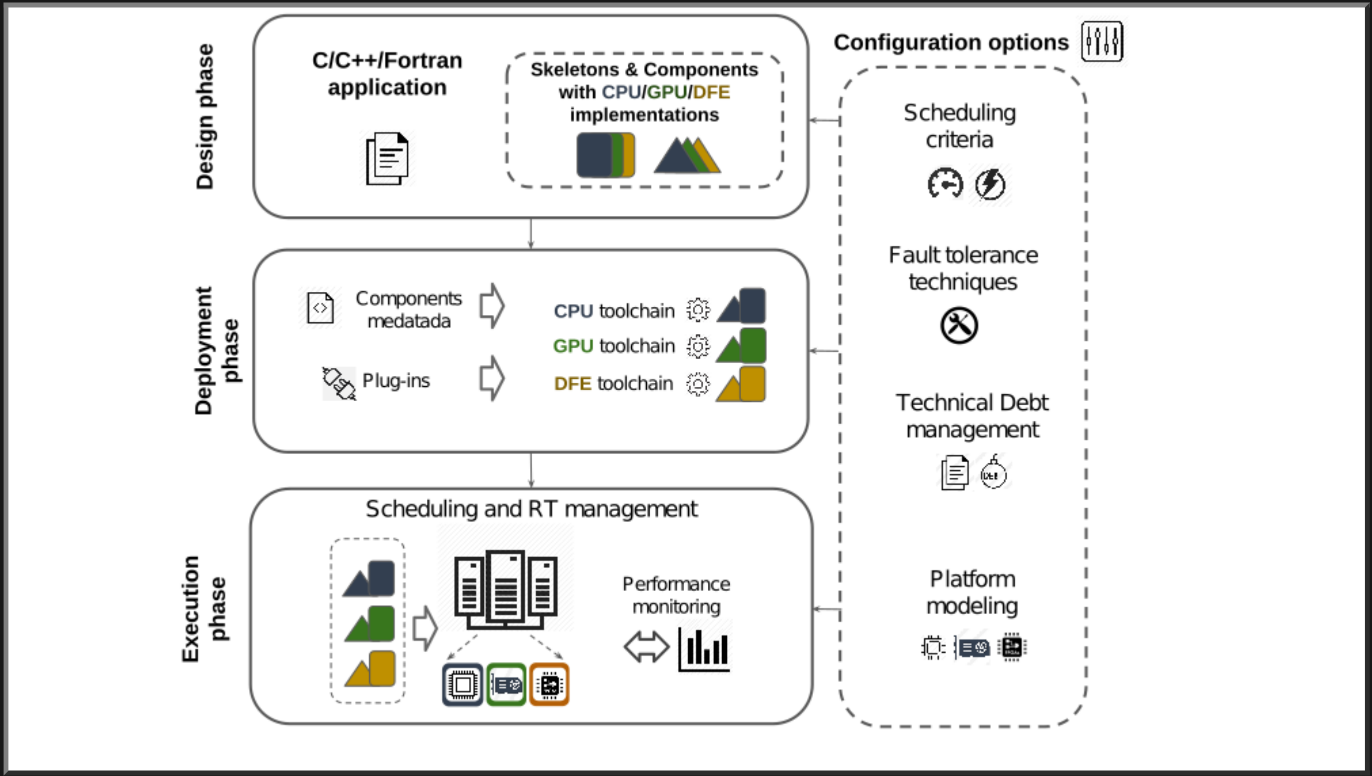
FPGA: a new kind of accelerator for scientific computing ?

Matthieu Haefele and Charles Prouveur

Café Calcul, June 2021

EXA2PRO H2020 FET-HPC project



Outline

- FPGA architecture and principles
- The MaxJ "programming" model and toolchain
- Porting MetalWalls mini-app on FPGA
- Number representation and numerical accuracy
- Comparison in time and energy with CPU and GPU
- Trying to look into the future...

You said FPGA ?

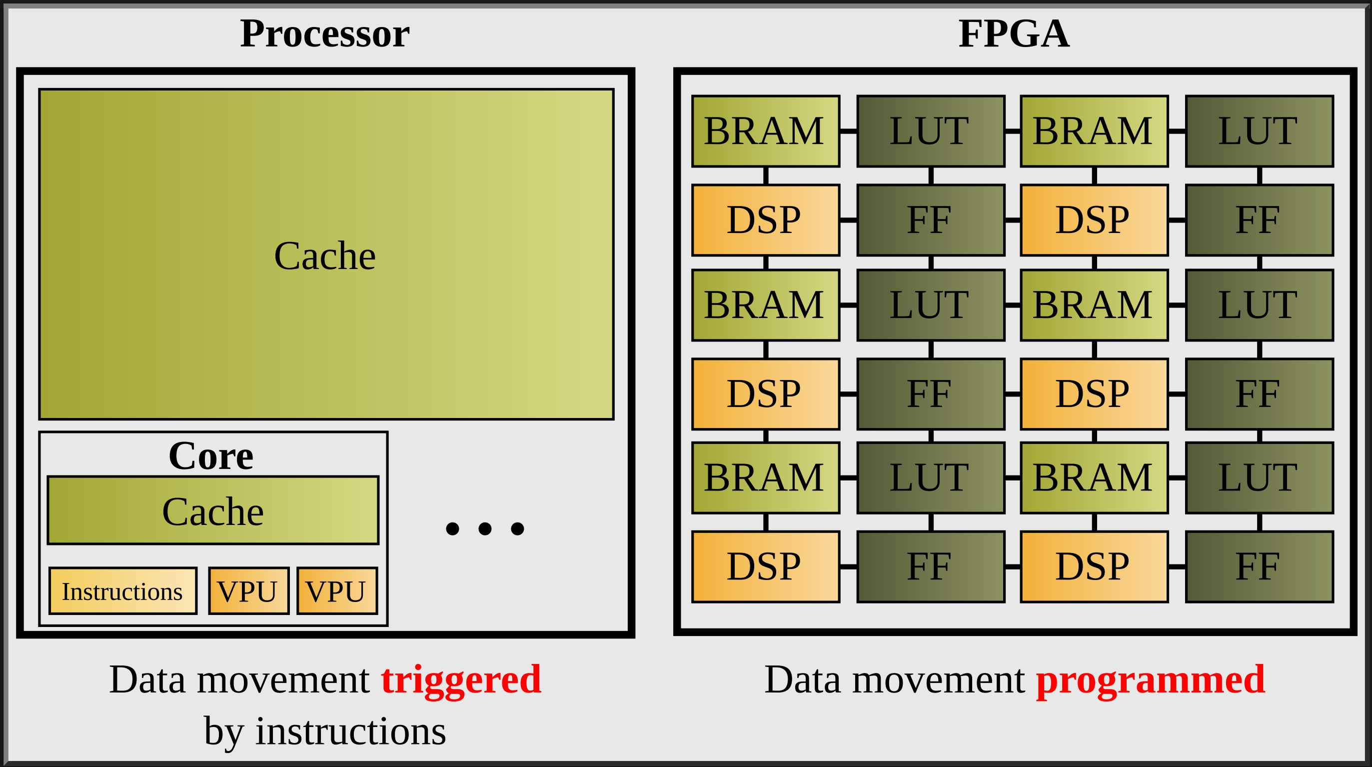
Some acronyms

- 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)

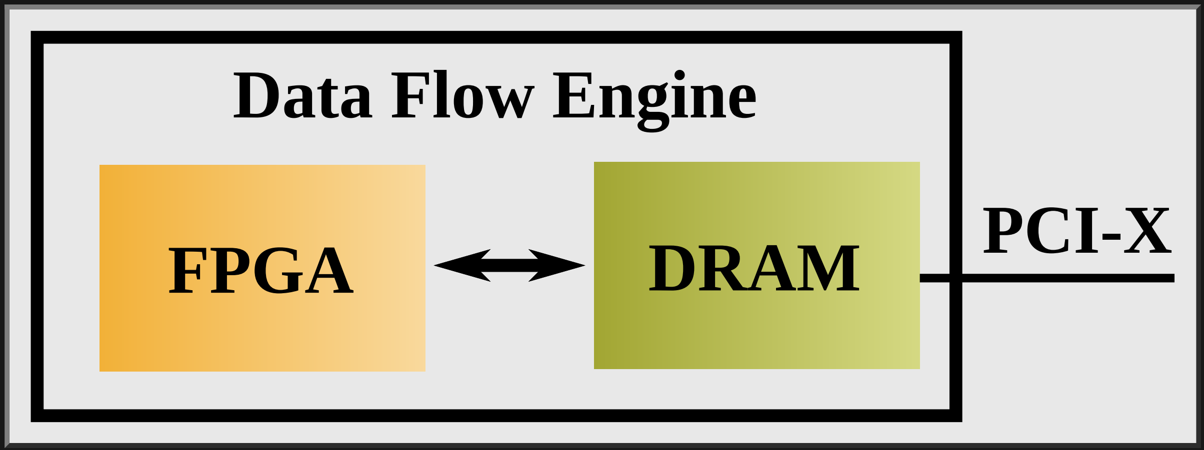
But what is it ?

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

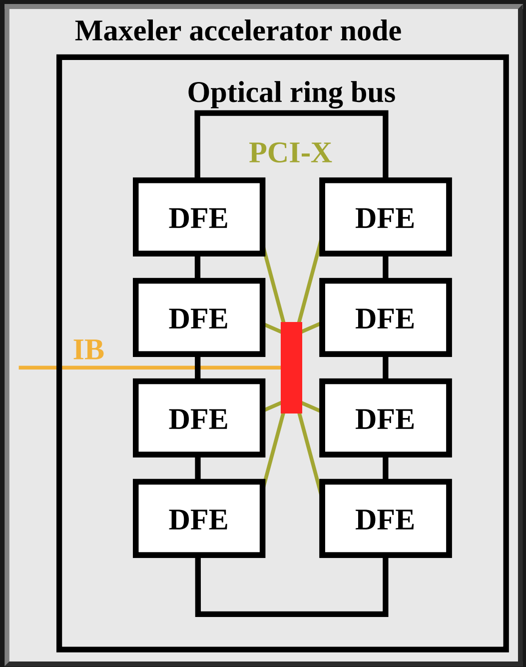
FPGA $\neq$ Processor



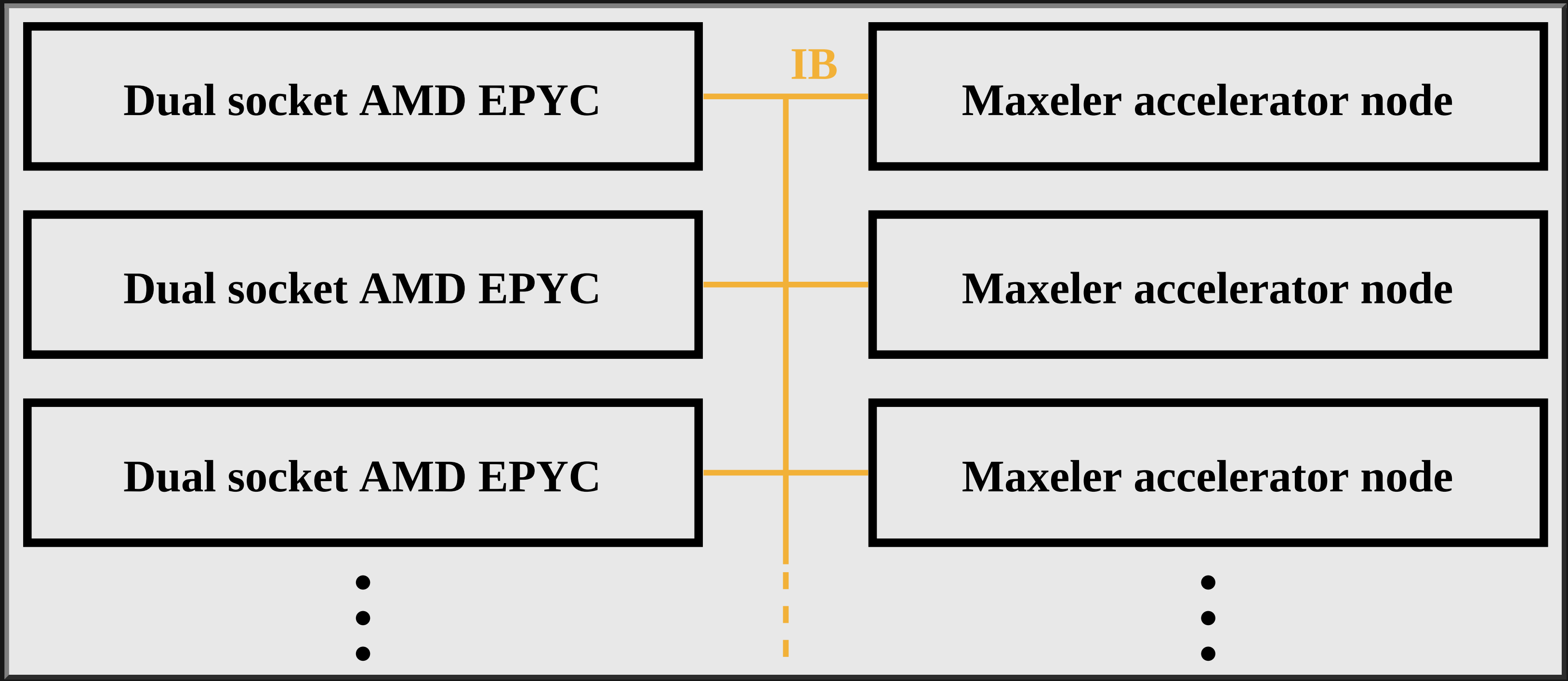
Maxeler Data Flow Engine (DFE)



Maxeler accelerator node



Computing system



Architecture comparison

Chip	Intel SkyLake	NVidia Pascal	Xilinx XCVU9P
Techno.	14nm	16nm	16nm
Power	205W	300W	**< 50W**
Freq.	2.7GHz	1.5GHz	**0.1-0.5GHz**
cache	57MiB	18 MiB	62 MiB
HBM / MCDRAM	0	16GB	0
DRAM	128-768 GB	0	48GB

"Programming" model

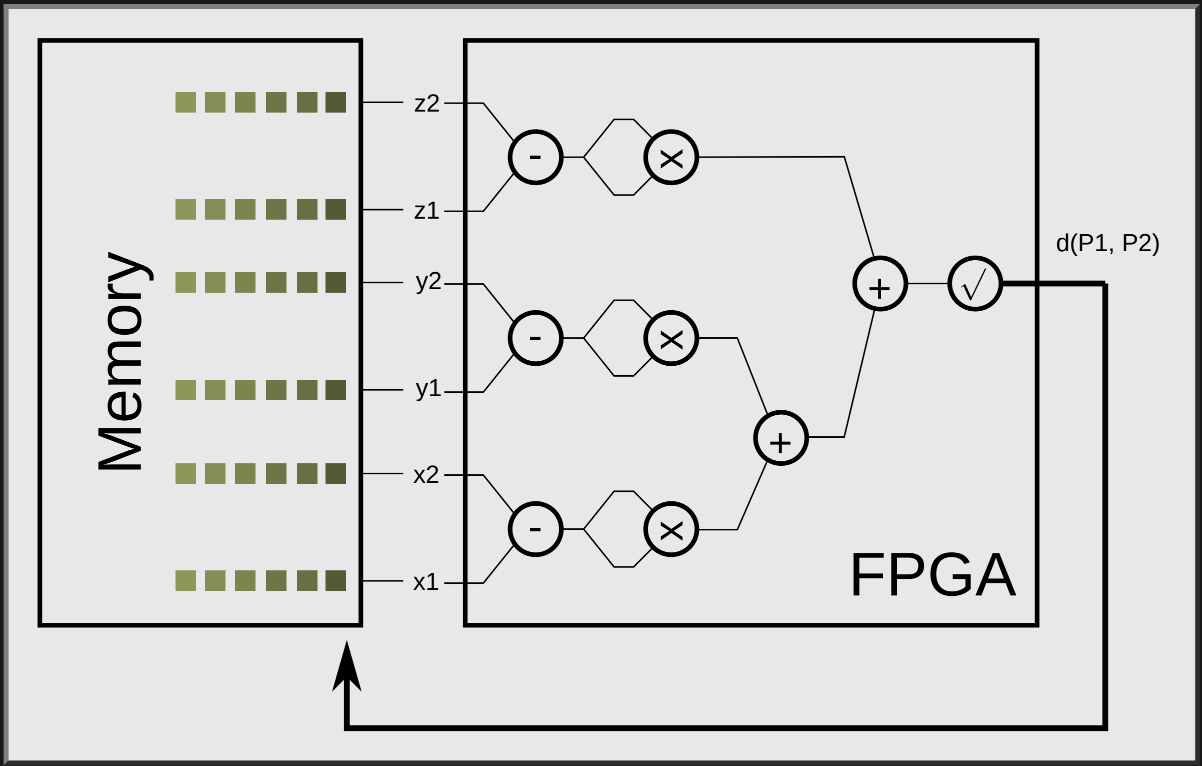
Available languages

- VHDL, the standard: very low level for electronic people
- XilinX HLS: Pragmas for C
- OpenCL: C-like offload API
- SycL or Intel OneAPI: C++ framework
- **Maxeler MaxJ**: Domain Specific Language based on Java
- XilinX Vitis: AI Development Environment

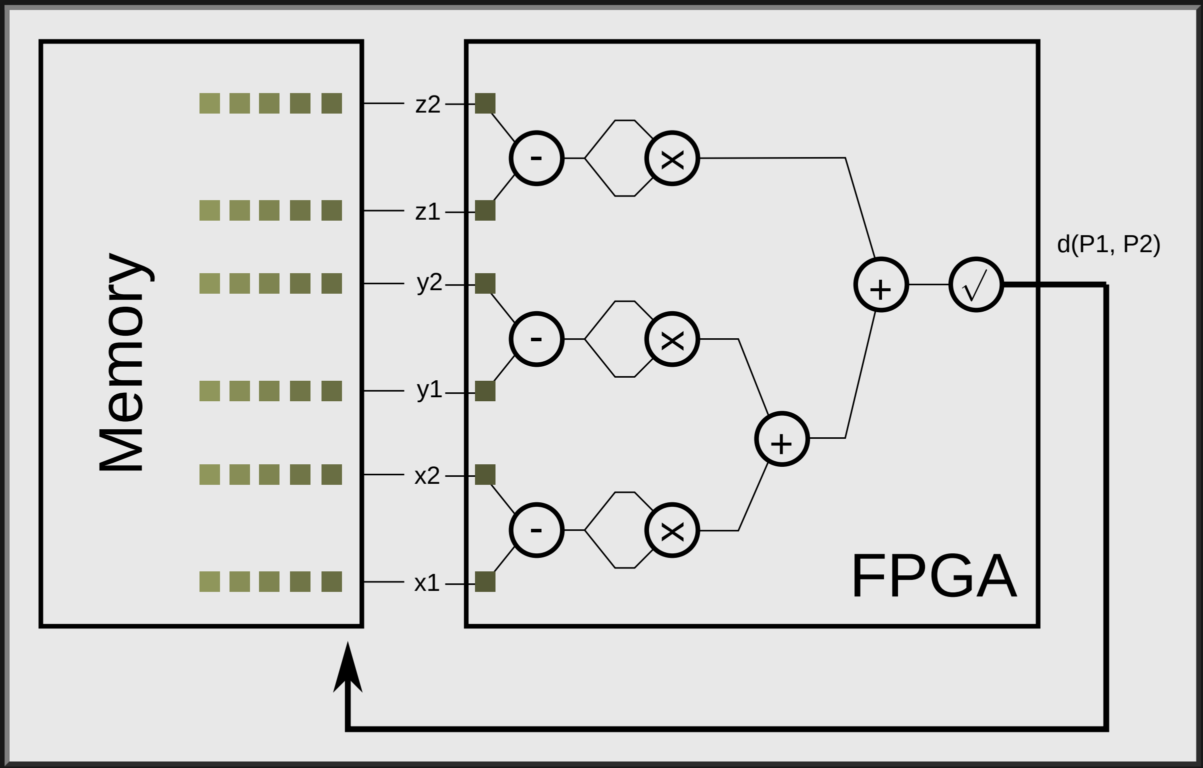
Challenges

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

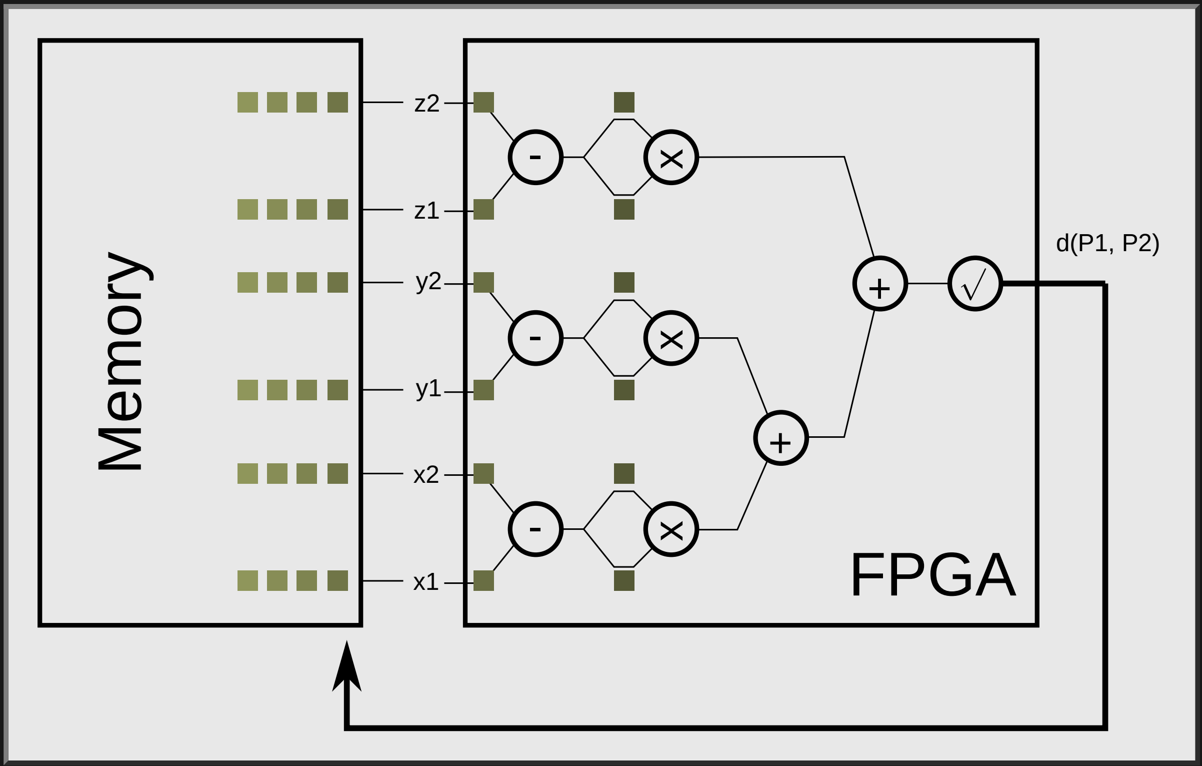
Example: computing a distance



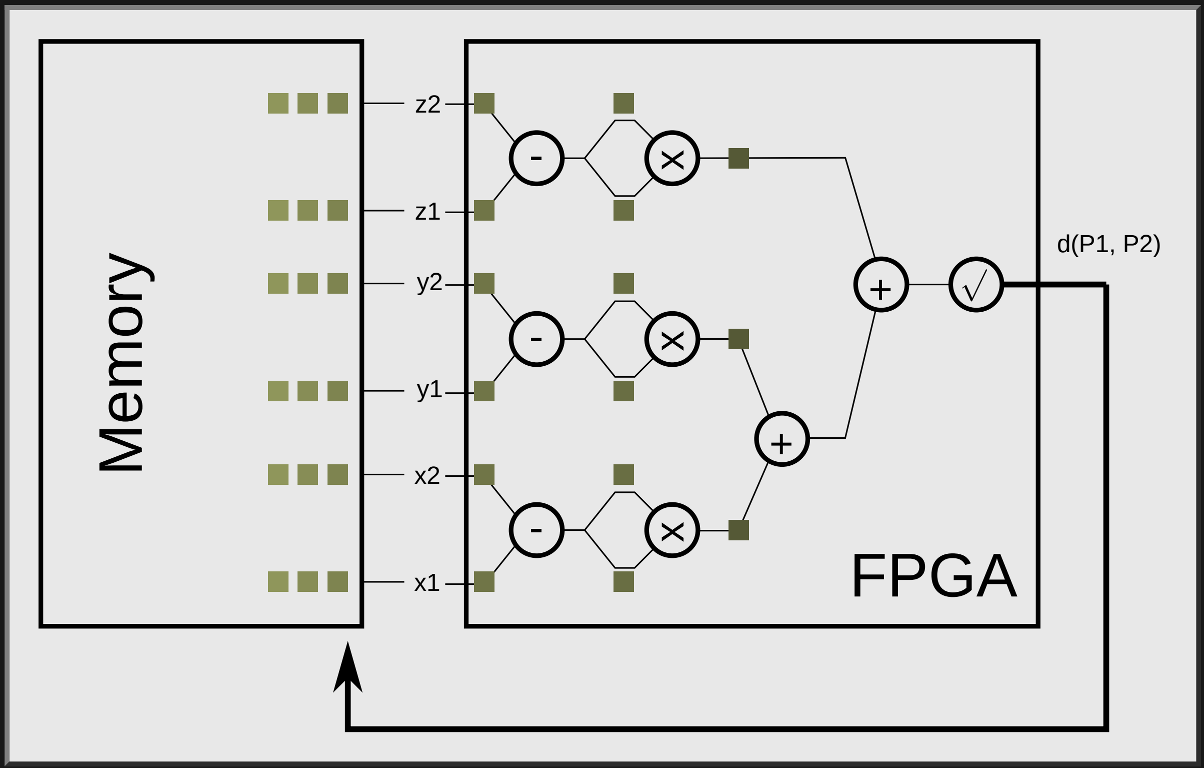
Example: computing a distance



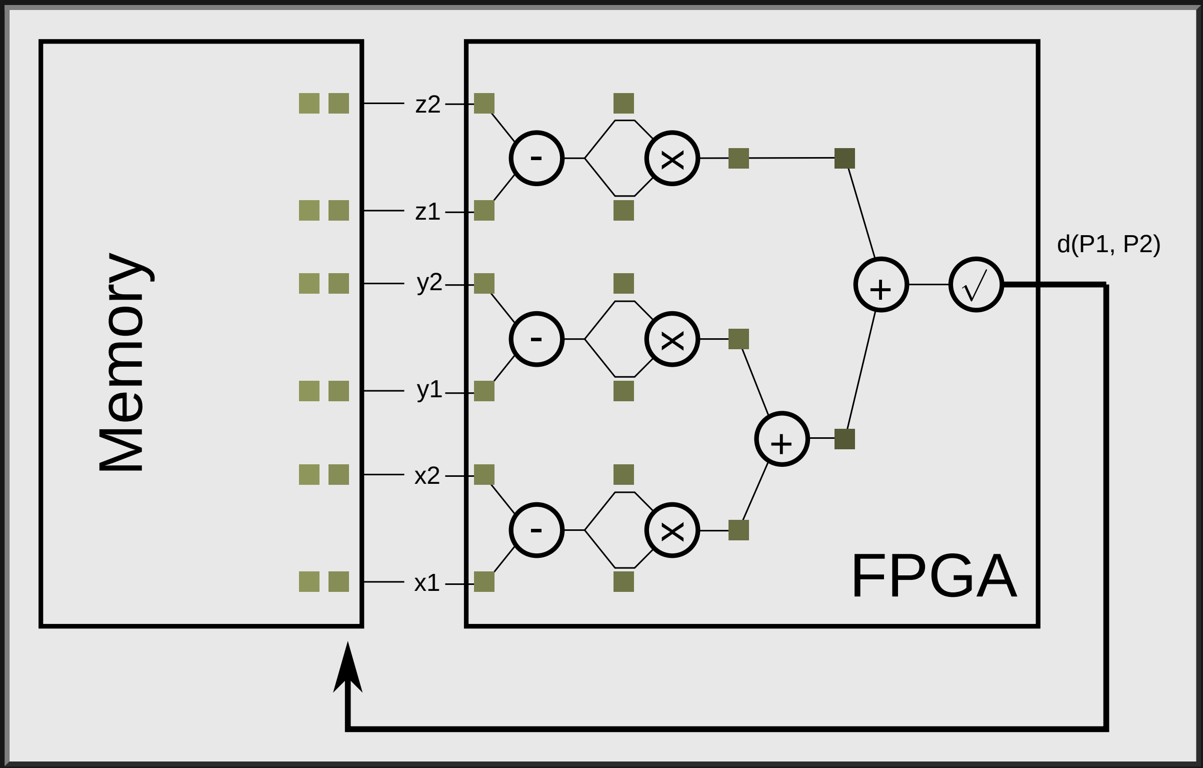
Example: computing a distance



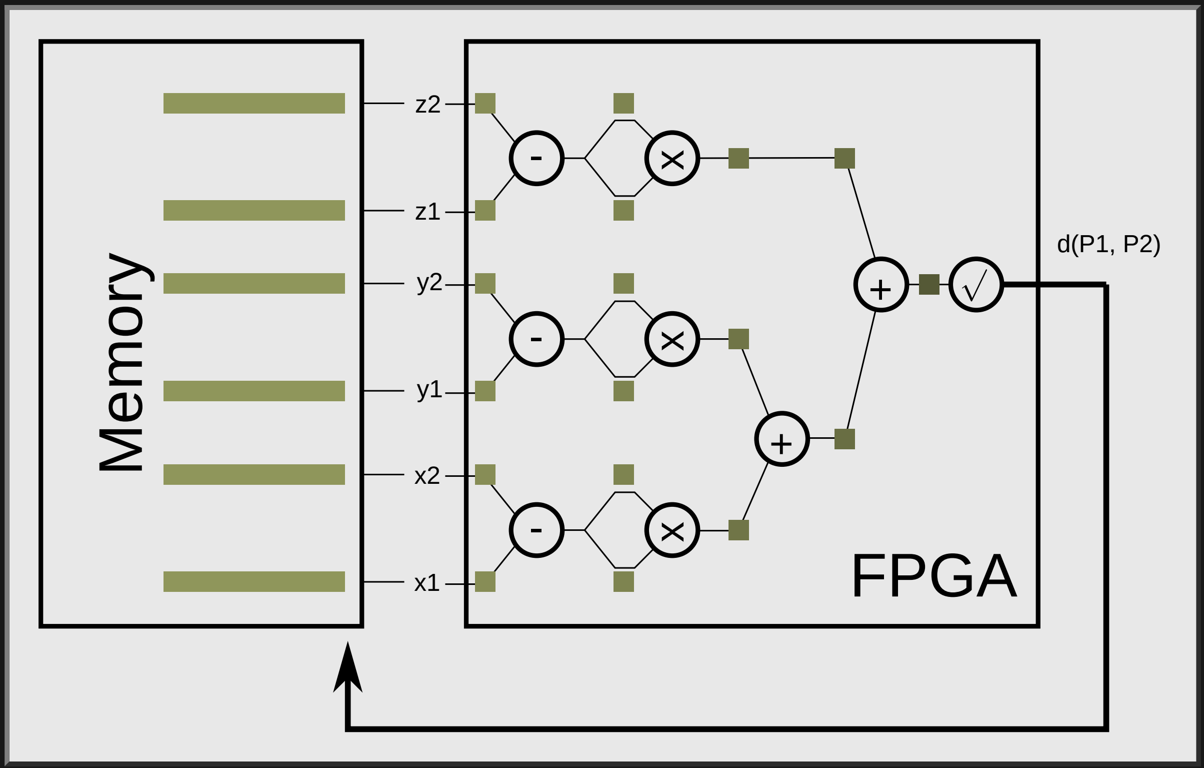
Example: computing a distance



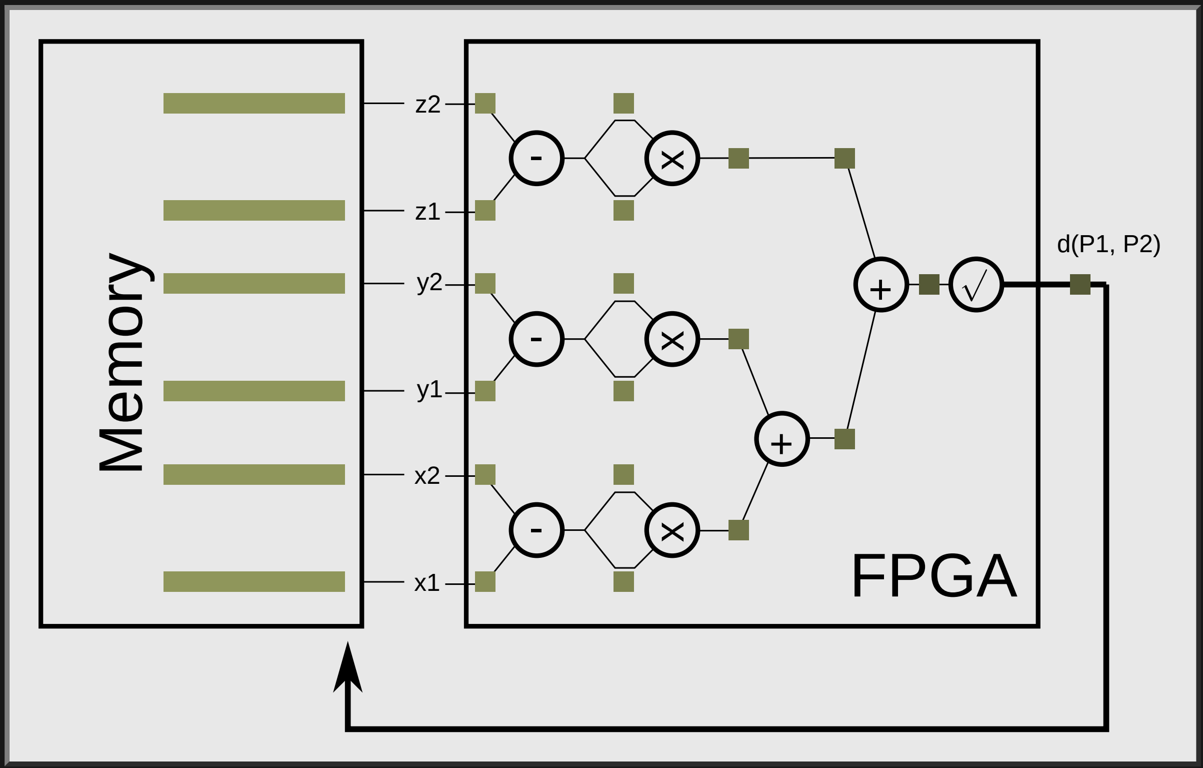
Example: computing a distance



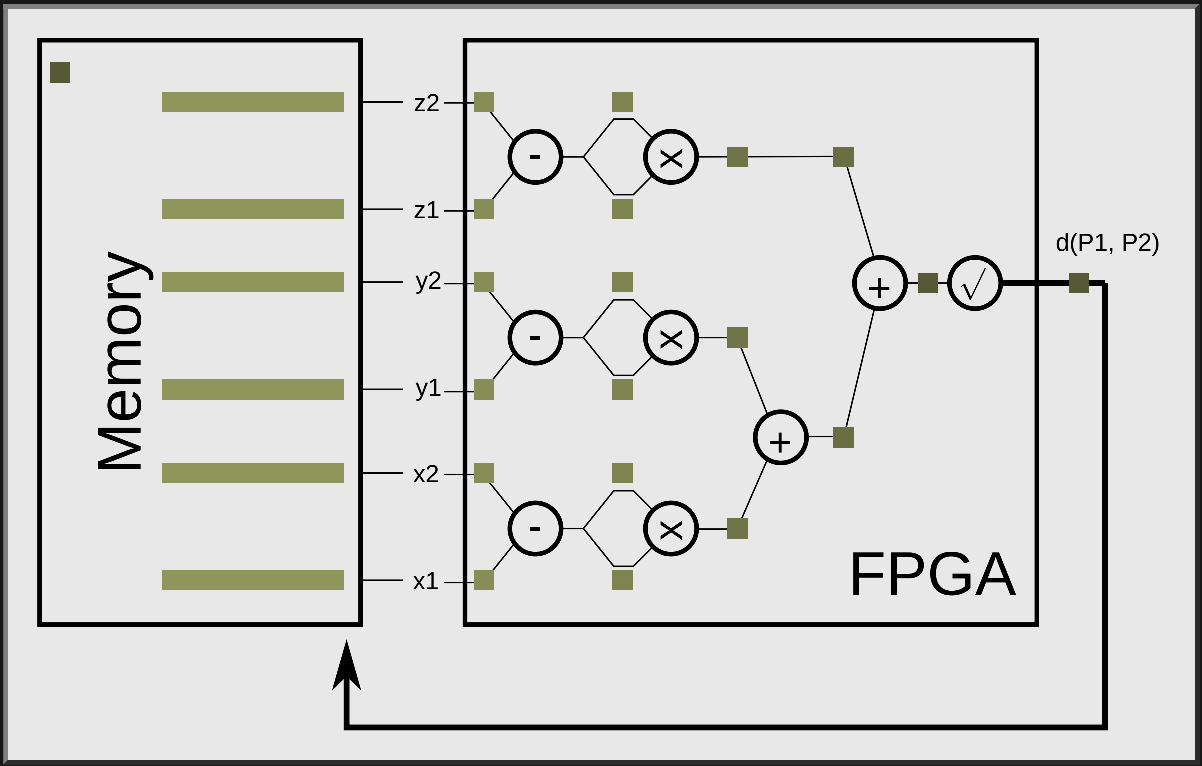
Example: computing a distance



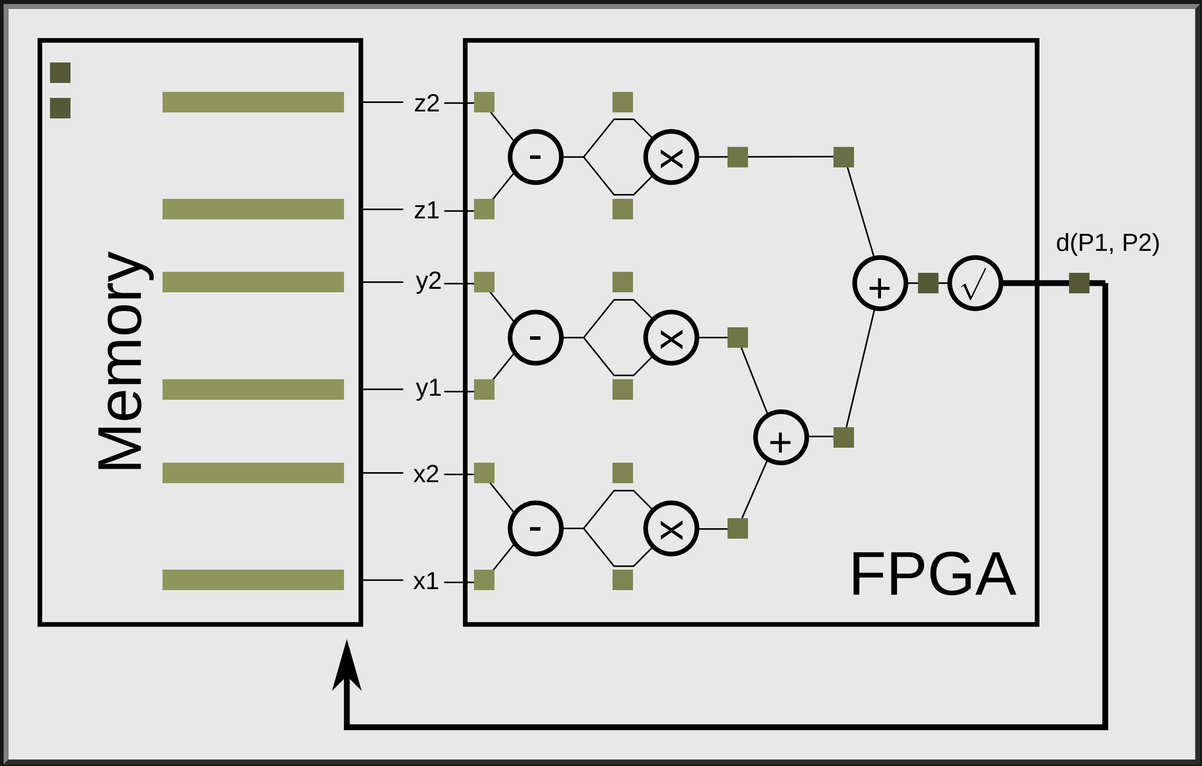
Example: computing a distance



Example: computing a distance



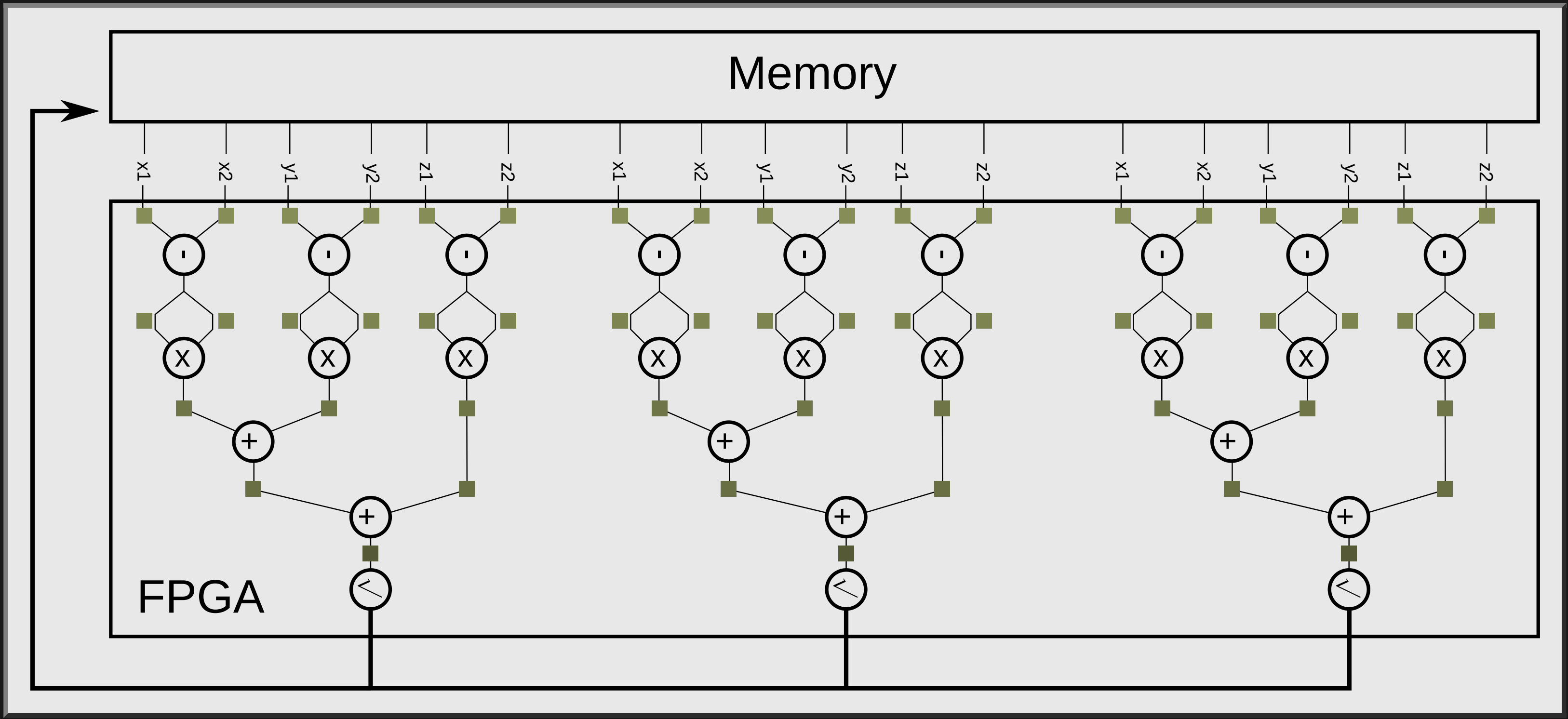
Example: computing a distance



Example: computing a distance

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

Example: computing a distance



Development workflow 1/3

- 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 2/3

- 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
- Algorithm correctness performed in simulator / emulator

Development workflow 3/3

- **24-48 hours needed for kernels compilation !!**
- Algorithm correctness checked on real hardware

Distance MaxJ kernel

```
public class distKernel extends Kernel {
  public distKernel(final KernelParameters parameters) {
    super(parameters);

    DFEType DP_float = dfeFloat(11, 53);
    DFEType distance = DP_float;
    DFEType point = new DFEVectorType<DFEVar>(DP_float, 3);

    DFEVar P1 = io.input("P1", point);
    DFEVar P2 = io.input("P2", point);

    DFEVar dx = P1[0] - P2[0];
    DFEVar dy = P1[1] - P2[1];
    DFEVar dz = P1[2] - P2[2];
    DFEVar d = sqrt(dx*dx + dy*dy + dz*dz);

    io.output("distance", d, distance);
}}
```

Distance MaxJ manager

```
public class distManager extends MAX5CManager {
  public distManager(EngineParameters params) {
    super(params);

    KernelBlock kernel = addKernel(
           new distKernel(makeKernelParameters("distKernel"))
           );

    DFELink P1 = addStreamFromCPU("P1");
    DFELink P2 = addStreamFromCPU("P2");
    kernel.getInput("P1") <== P1;
    kernel.getInput("P2") <== P2;

    DFELink distance = addStreamToCPU("distance");
    distance <== kernel.getOutput("distance");
  }
}
```

Distance main.c 1/2

```
#include "distance.max"
#include "distance.h"
#define N 16

int main(int argc, char** argv)
{
  double P1[3*N], P2[3*N], distance[N];
  max_file_t    *maxfile = distance_init();
  max_engine_t  *engine  = max_load( maxfile, "*" );

  init(P1, P2);
  distance_DFE(maxfile, engine, P1, P2, distance);
  print(distance);
  return 0;
}
```

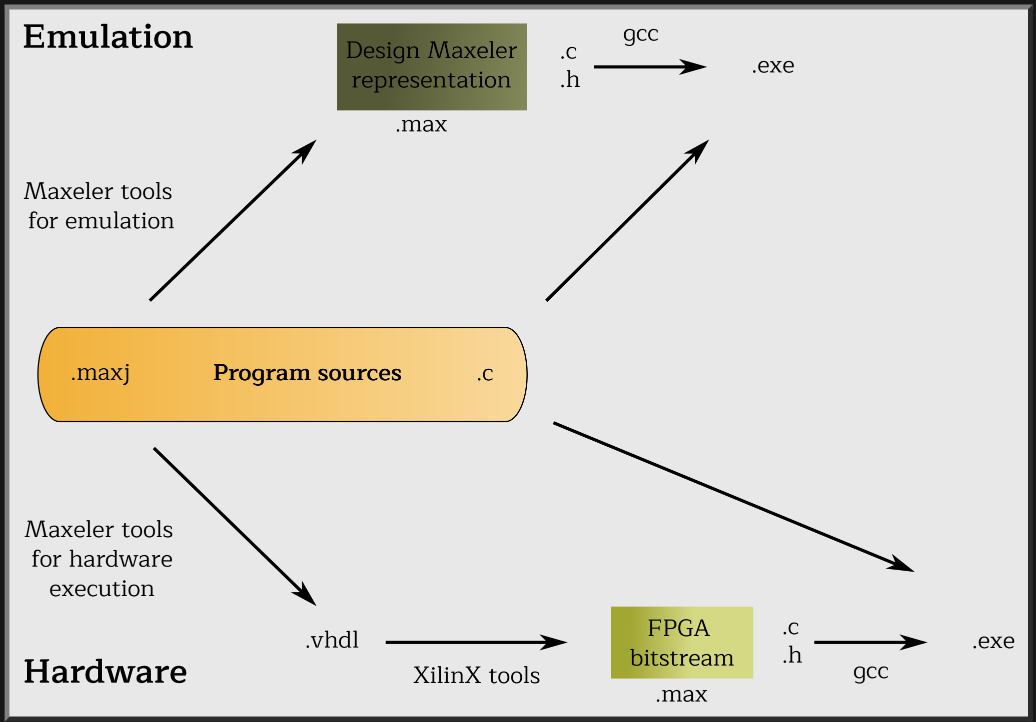
Distance main.c 2/2

```
void distance_DFE(max_file_t* maxfile, max_engine_t* engine, \
                  double* P1, double* P2, double* distance)
{
  max_actions_t* act = max_actions_init(maxfile, NULL);
  max_queue_input(act, "P1", P1, 3*N*sizeof(double));
  max_queue_input(act, "P2", P2, 3*N*sizeof(double));
  max_queue_output(act, "distance", distance, N*sizeof(double))

  max_set_ticks(act, "distKernel", N);

  max_run(engine, act);
  max_actions_free(act);
}
```

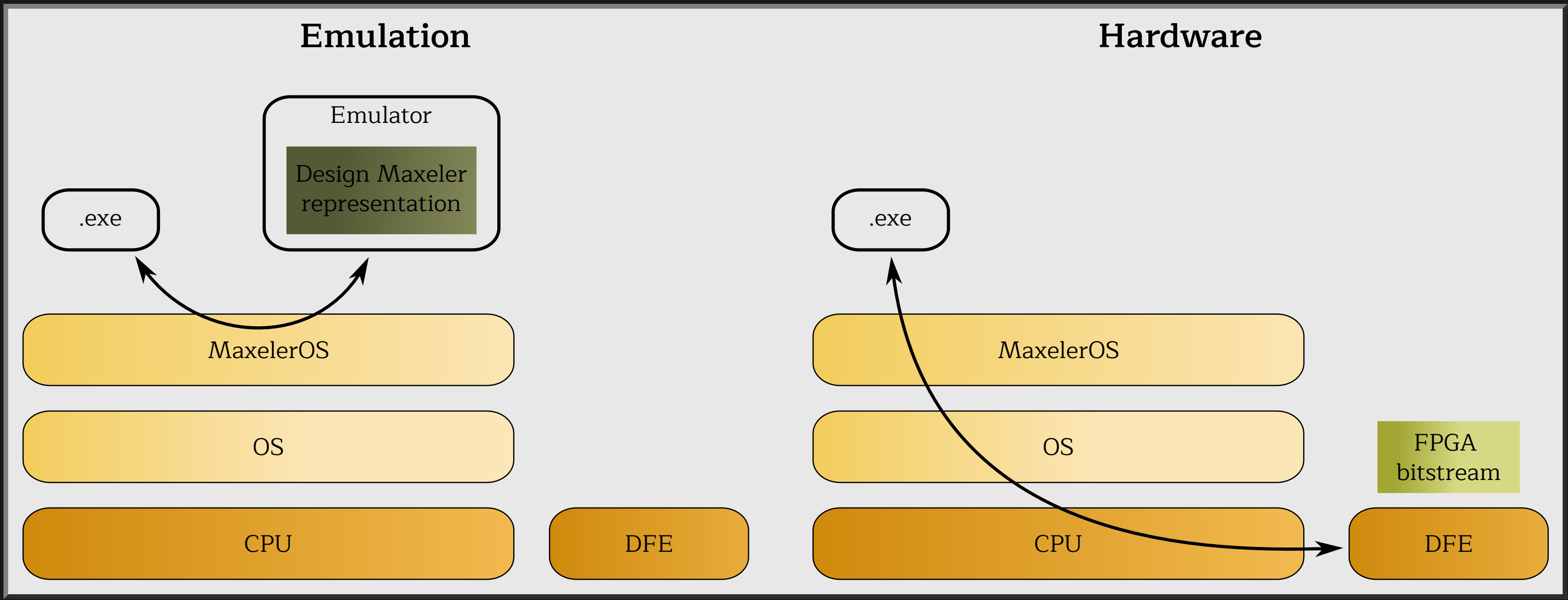
Distance MaxJ kernel multi-pipe

```
public class distKernel extends Kernel {

  public distKernel(final KernelParameters parameters, int numP
    super(parameters);

    ...
    DFEType point = new DFEVectorType<DFEVar>(DP_float, 3*numPi
    DFEVar P1 = io.input("P1", point);
    DFEVar P2 = io.input("P2", point);

    for(int i=0; i< numPipe ; i++){
      dx[i] = P1[0 + 3*i] - P2[0 + 3*i];
      dy[i] = P1[1 + 3*i] - P2[1 + 3*i];
      dz[i] = P1[2 + 3*i] - P2[2 + 3*i];
       d[i] = sqrt(dx[i]*dx[i] + dy[i]*dy[i] + dz[i]*dz[i]);
    }
    io.output("distance", d, distance);
}}
```

At compile time



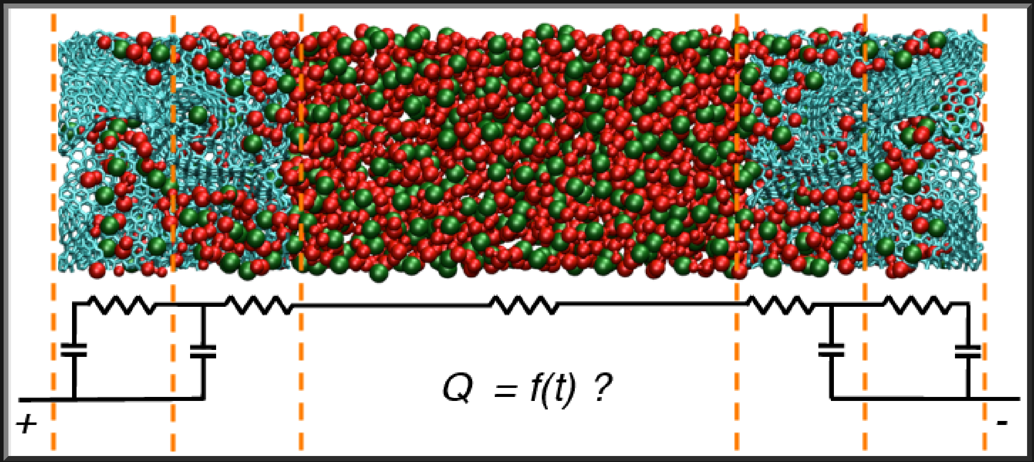
At runtime



Emulation
Hardware
Emulator
Design Maxeler representation
.exe
MaxelerOS
OS
CPU
DFE
.exe
MaxelerOS
OS
FPGA bitstream
CPU
DFE

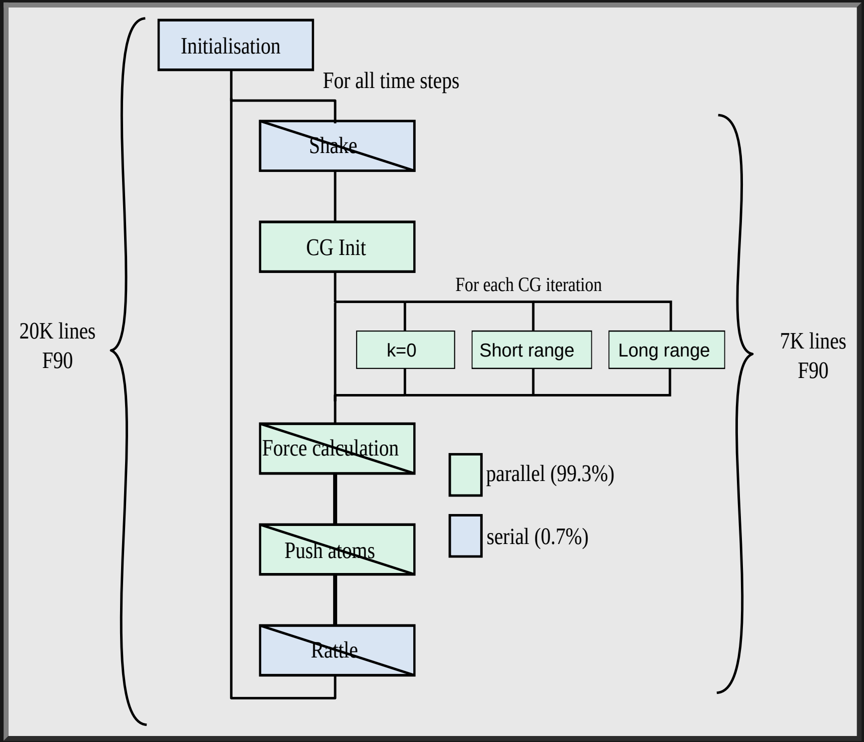
MetalWalls

- Molecular dynamics with accurate electrostatic
- Simulation of electrochemical systems
- Developed by M. Salanne (Sorbonne University)



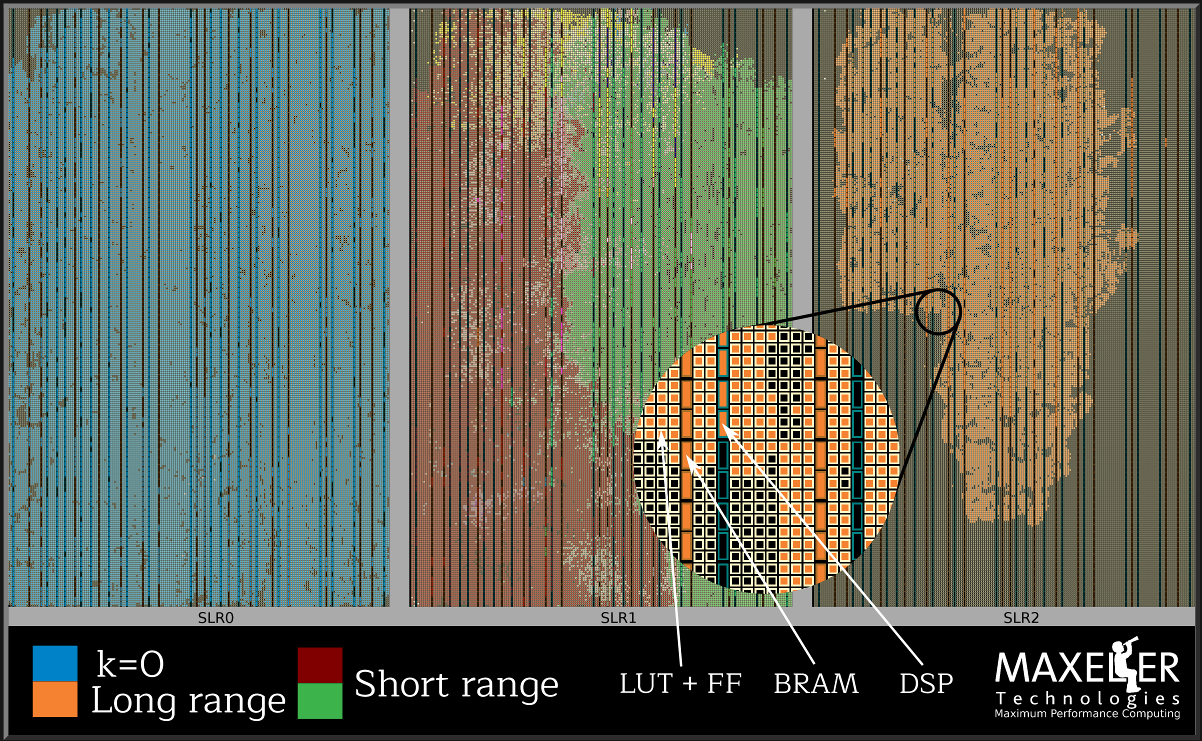
- Carbon electrodes (Blue), Ionic liquid (red/green)
- Study of the liquid/electrode interface

Algorithm



Initialisation
For all time steps
Shake
CG Init
For each CG iteration
k=0
Short range
Long range
Force calculation
Push atoms
Rattle
parallel (99.3%)
serial (0.7%)
20K lines
F90
7K lines
F90

DFE implementation



DFE porting effort

- Two weeks intensive support at Maxeler
- 6 months learning + first implementation
- $\Rightarrow$ First results in emulation
- 6-8 months to get an efficient design on the hardware

Number representation

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

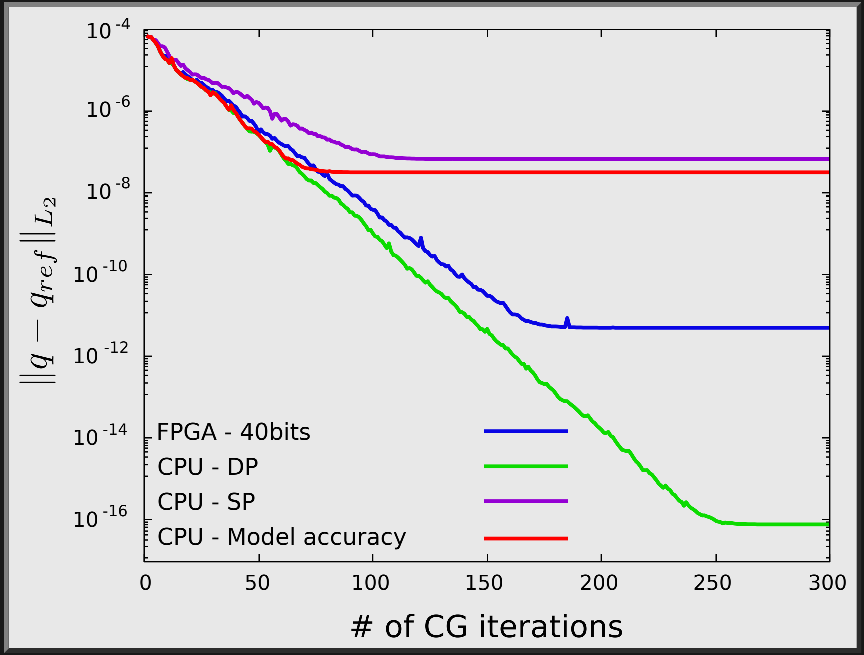
- Any floating point representation available, even fixed point
- Requires less resources (2x - 6x between SP and DP)

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

Numerical accuracy analysis

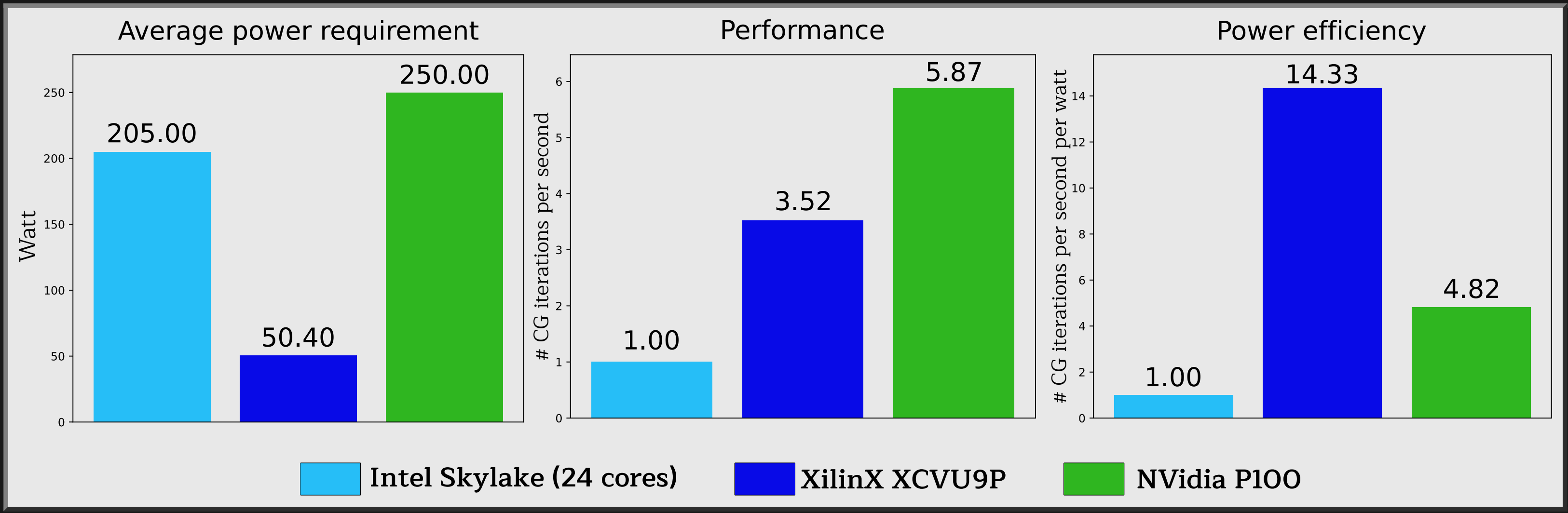
- r_{cut} parameter of the Ewald summation
- For a given r_{cut}, runs performed with QP, DP, SP and exotic FPGA precisions
- $\Rightarrow$ Norm of difference of result reflects number representation error
- Runs with DP for different r_{cut}
- $\Rightarrow$ Norm of difference of result reflects model accuracy

40 bits are enough !

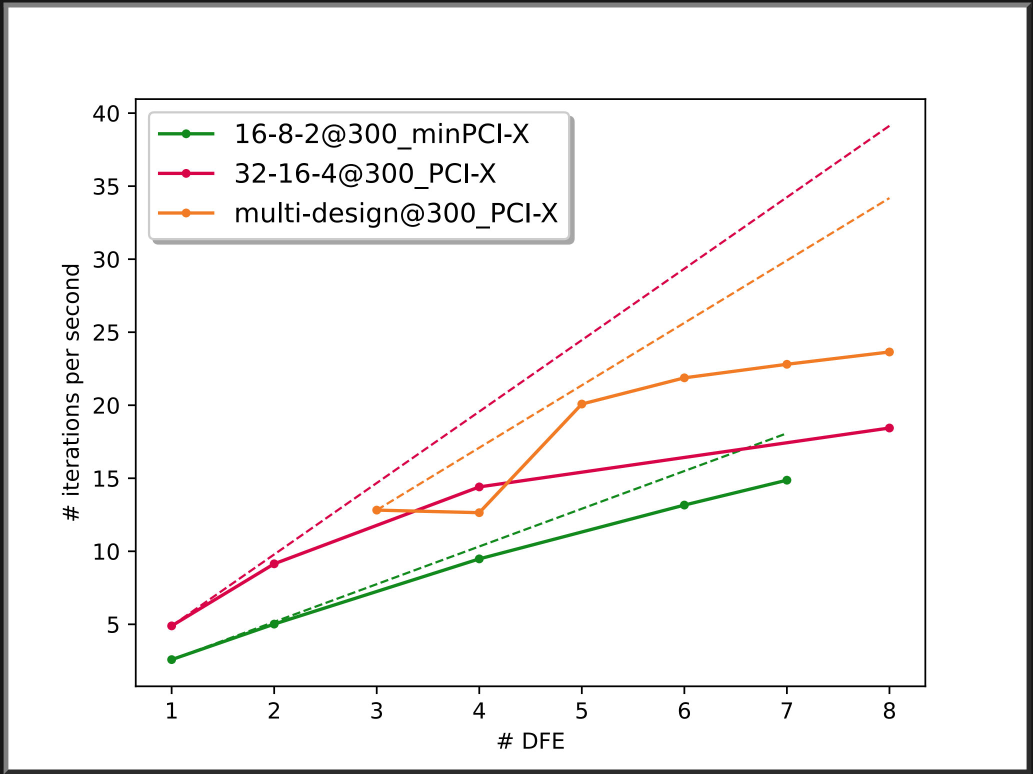


CADNA able to find the model accuracy without QP run.

CPU vs FPGA vs GPU



Multi-DFE implementation



Conclusion

- DFE programming model is an appropriate level of abstraction
- Flat learning curve and sparse documentation for HW generation
- For the MetalWalls case:
  - 40 bits are enough to represent numbers in Metalwalls
  - FPGA x2 slower than GPU
  - FPGA x3 more power efficient than GPU

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

Looking into the future

- Moore's law ending in a near future
  - $\Rightarrow$ Architecture is likely to become the driver for more performance
- Intel bought Altera in 2015 for 16.7 B$
- AMD announced to buy XilinX in 2020 for 35 B$
  - $\Rightarrow$ CPUs will likely integrate reconfigurable logic
  - $\Rightarrow$ FPGAs are integrating more and more specialised blocks

HPC might have to deal with this reconfigurable logic at some point