Imperial College London

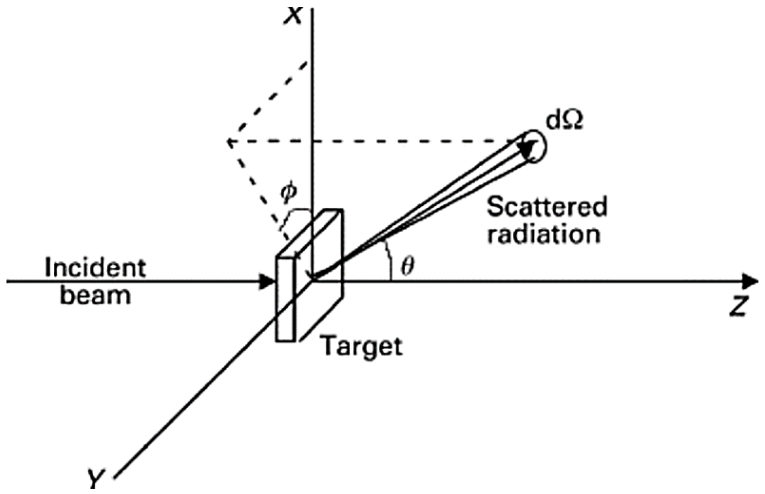
Demonstration of FPGA Acceleration of Monte Carlo Simulation

Marco Barbone
m.barbone19@imperial.ac.uk

Imperial College London

Elastic scattering

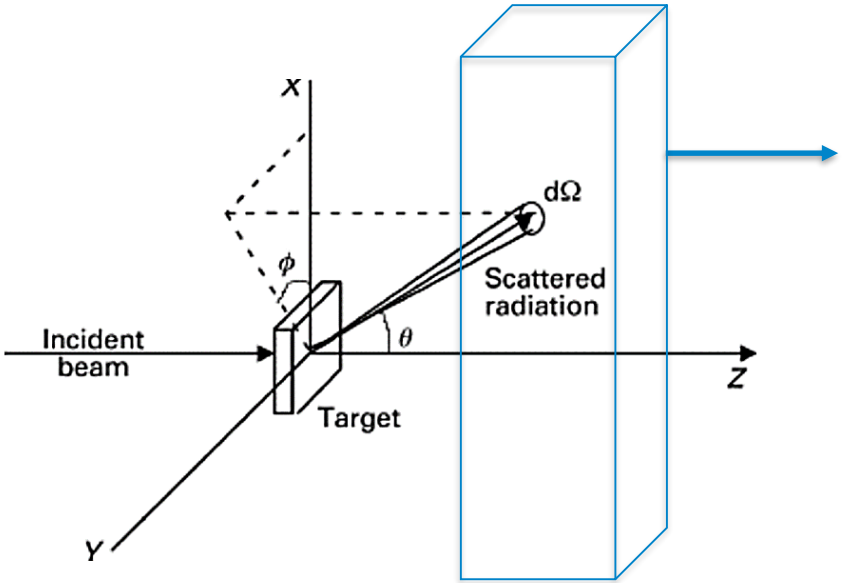
A simulation of e-/e+ transport considering only elastic scattering as possible interactions described by scattering of spin-less e-/e+ on an exponentially screened Coulomb potential



Source: Kotlarchyk M. Scattering theory. Encyclopedia of Spectroscopy and Spectrometry, 1999.

Imperial College London

Monte Carlo Simulation



Scattered radiation final position

Many particles are simulated to achieve statistical significance

Source: Kotlarchyk M. Scattering theory. Encyclopedia of Spectroscopy and Spectrometry, 1999.

Imperial College London

FPGA acceleration

Imperial College London

Methodology

No agile development! Compilation might take days

A methodology is needed to minimize unneeded compilations:

1. Application analysis
2. Performance and resource modelling
3. Decide the acceleration target
4. Model the target in software
5. FPGA programming

Imperial College London

Workload selection

Not all workloads benefits from FPGA acceleration!

FPGA have higher throughput than CPUs (and GPUs) if the workload:

- Has many branch mispredictions
- Has many cache faults
- (optional) Embarrassingly parallel
- (optional) Independency

→ Monte Carlo fits all!

Imperial College London

FPGA Performance

- FPGA Performance is predictable
- There is no context switch, garbage collector or any background process
- The bitstream will be executed the same number of clock cycles every time
- The number of clock cycles needed can be computed easily

Further read: Nils Voss et al. (2021), On Predictable Reconfigurable System Design

https://doi.org/10.1145/3436995

Imperial College London

Requirements

We assume that FPGA should achieve 10x speed-up (over a modern parallel CPU) to be worth the engineering-effort

- The simulation of one particle requires 34.5 iterations on average
- Assuming an FPGA architecture capable of computing 1 iteration per clock cycle to achieve a 10x speed-up the FGPA clock frequency should be 250Mhz
- Alternatively, a parallelism > 1 at a lower clock frequency

Imperial College London

Model findings

Parallelism = 1

- DSP utilization = 127
- On-chip memory requirement = 10Mb

The VU9P FPGA used in this study has 6840 DSP and 345Mb on-chip memory

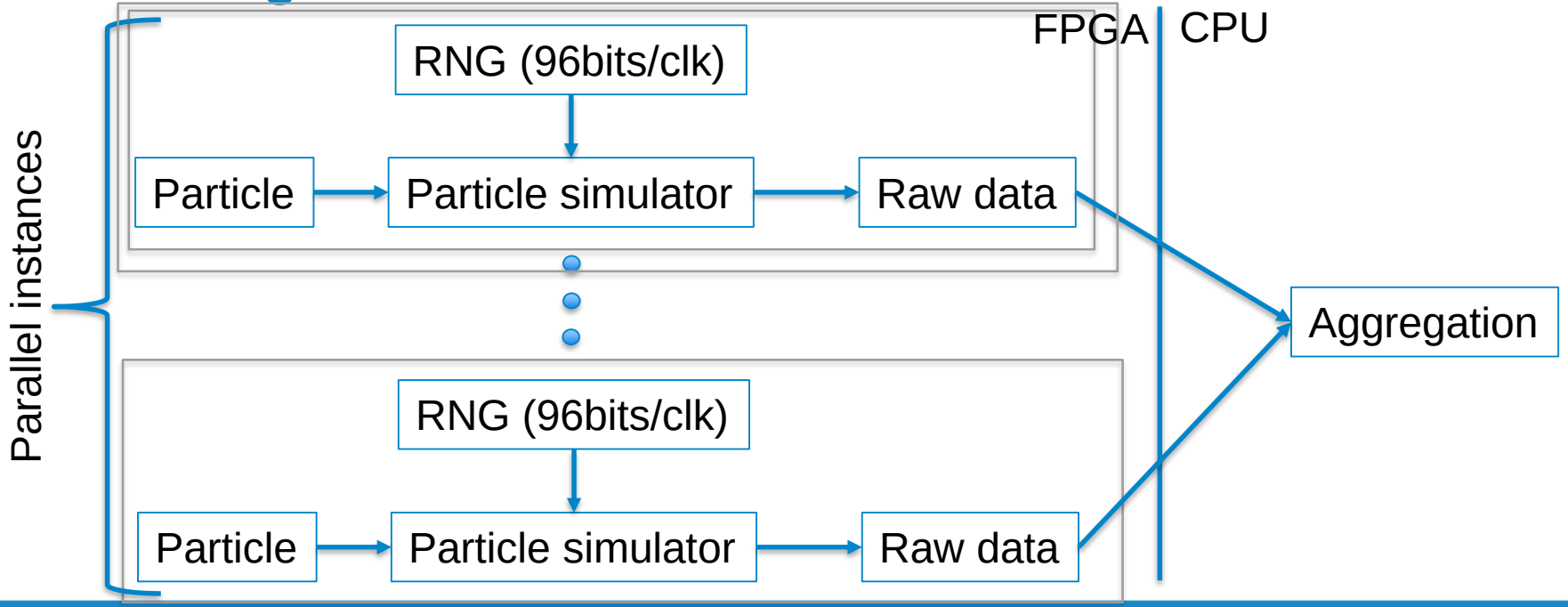
The architecture requires 1.86% DSPs and 3.35% on-chip memory resources

Limiting factor

Theoretical parallelism while aiming at consuming at most 50% of the on-chip memory resources is 15

Imperial College London

Resulting architecture



Imperial College London

Measurement results

Parallelism = 15

- DSP utilization = 1904 (27%)
- On-chip memory requirement = 300Mb (86%) (scheduling overhead)

Higher parallelism could limit the achievable clock frequency

Imperial College London

Test configurations

Hardware:

- AMD Ryzen 5900x 12-cores 3.7GHz (4.8GHz) CPU
- Xilinx's Alveo U200 Data Center accelerator card Xilinx VU9P FPGA

Toolchain:

- MaxCompiler 2021.1
- Vivado 2019.2
- OpenMP

Imperial College London

Test configurations (continued)

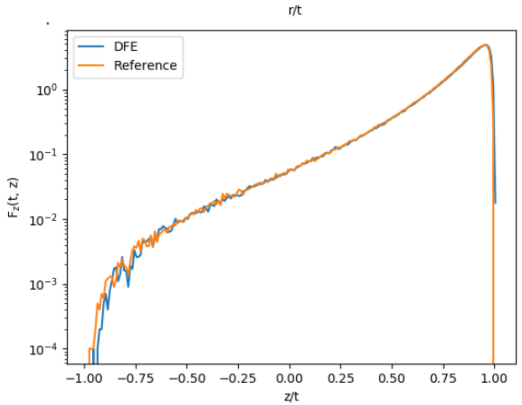
Simulation configuration:

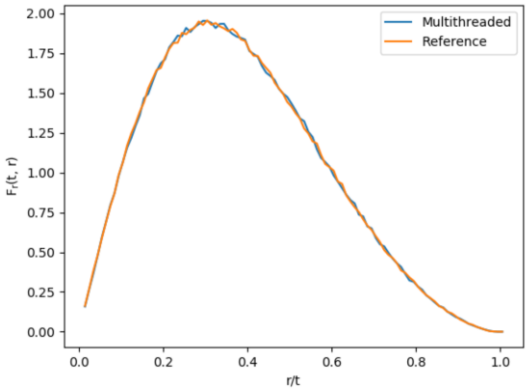
- 100M histories
- 6 MeV beam
- Water

The FPGA clock frequency was limited to 200MHz

Imperial College London

Histograms





It passes a KS-test

Imperial College London

Performance measurements

The FPGA accelerated version is:

- 270x faster than an optimized single-core implementation
- 110x faster than multi-core implementation (12-cores)
- 10x cost-equivalent speedup

The theoretical 128x does not account for overhead and CPU data aggregation

Using MaxCompiler the architecture achieves 300Mhz resulting in a 160x speedup

Imperial College London

Conclusions

- FPGAs can be multiple orders of magnitude faster than CPUs and GPUs when accelerating suitable workloads
- Due to their stochastic nature Monte Carlo simulations greatly benefit from FPGA acceleration
- The results shows 110x speedup compared to high-end multicore consumer CPUs
- Following a formal methodology and HLS toolchains high performance can be achieved with no previous FPGA programming experience

Imperial College London

Worst-case for branch predictors

Assuming rng uniformly distributed:

```
if(rng.getRandom() < 0.5) then f(x) else g(y)
```

In the GPU case branches can significantly affect the instruction throughput by causing threads of the same warp to diverge.

The diverging branches are de-scheduled!