MATENVMED : ΔΡΑΣΗ 3.2

ΥΛΟΠΟΙΗΣΗ ΣΕ FPGAs ΚΑΙ ΠΟΛΥΠΥΡΗΝΑ ΣΥΣΤΗΜΑΤΑ

Νικόλαος Μπέλλας, Χρήστος Αντωνόπουλος,
Βασίλης Βασιλειάδης, Γεώργιος Ζήνδρος, Μανώλης Μαρούδας
Πανεπιστήμιο Θεσσαλίας

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

1. DELIVERABLE 3.2

- Select and study computationally demanding kernels of WP2 that can potentially be accelerated when mapped to a massively parallel platform (GPUs, FPGAs).
- Design and implement appropriate tools or utilize third-party tools to help with mapping these kernels to such platforms (compilers, CAD tools)
- Map the kernels to FPGAs and GPUs
- Performance analysis and comparison wrt. to CPU implementation

2. FPGA TECHNOLOGY

•*Field Programmable Gate Array (FPGA)* is the best known example of Reconfigurable Logic

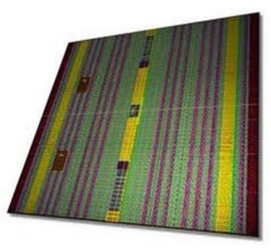
•Hardware can be modified post chip fabrication

•Tailor the Hardware to the application

–Fixed logic processors (CPUs/GPUs) only modify their software (via programming)

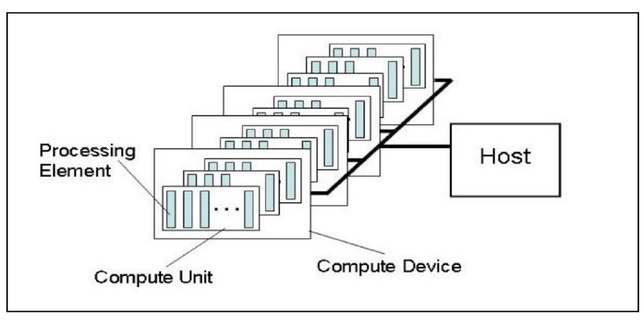
•FPGAs can offer superior performance, performance/power, or performance/cost compared to CPUs and GPUs.

- Advantages
  - Hardware tailored to application: potential for (near) optimal performance for a given application
  - Various forms of parallelisms can be exploited
- Disadvantages
  - –Programmable mainly at the hardware level using Hardware Description Languages (BUT, this can change)
  - –Lower clock frequency (< 300 MHz) compared to CPUs (~ 3GHz) and GPUs (~1.5 GHz)



3. OPENCL PROGRAMMING MODEL

- OpenCL (Open Computing Language) : A unified programming model aims at letting a programmer write a portable program once and deploy it on any heterogeneous system with CPUs and GPUs.
- Became an important industry standard after release due to substantial industry support



- One host and one or more Compute Devices (CD)
  - Each CD consists of one or more Compute Units (CU)
  - Each CU is further divided into one or more Processing Elements (PE)

OPENCL VECTOR ADD EXAMPLE

- OpenCL kernel describes the computation of a work-item (e.g. add two integer vectors)
- Each thread executes an instantiation of the OpenCL kernel code with different *idx*.

```
void add(int* a,
         int* b,
         int* c) {
for (int idx=0; idx<sizeof(a); idx++)
  c[idx] = a[idx] + b[idx];
}
```

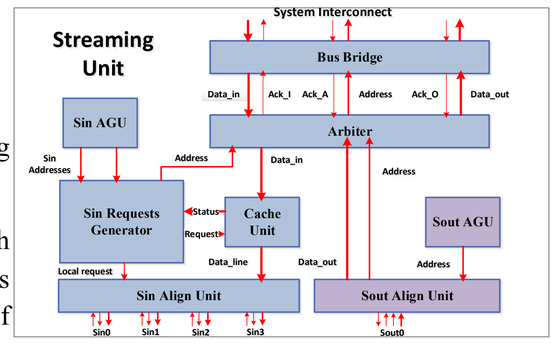
C code

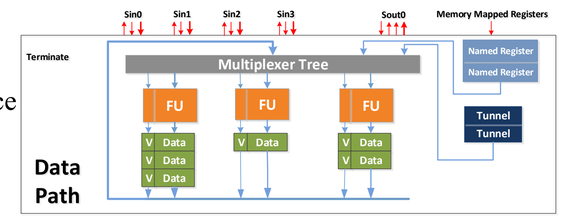
```
__kernel void vadd(
        __global int* a,
        __global int* b,
        __global int* c) {
    int idx= get_global_id(0);
    c[idx] = a[idx] + b[idx];
}
```

OpenCL kernel code

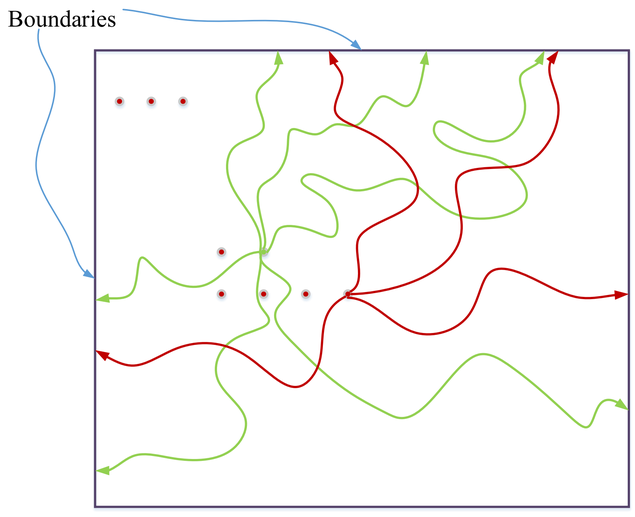
2. Silicon OPENCL

- OpenCL can be used as a CAD tool to generate hardware accelerators
- Fast hardware generation and architectural exploration using SOpenCL
- Common representation : one OpenCL program, multiple target platforms (CPU, GPU, FPGA)
- Generate configurable streaming architectures using OpenCL
- Discrete Data Path and Stream I/O units for separation of concerns
- Instruction clustering to reduce interconnect overhead





5. MONTE CARLO RANDOM WALKS



- All random walks from all points are independent. Each walk is sequential.
- Heavy-duty double precision arithmetic with non-constant number of iterations
- Compute-bound algorithm

$$do\{ rad = rand\_uniform(seed) * d; \}while(\frac{4 * rad}{(d * d)} * log(\frac{d}{rad}) < \frac{4 * rand\_uniform()}{e * d});$$

6. MANYCORE IMPLEMENTATIONS

- Our plan is to use the same code base (e.g. OpenCL) to explore different architectures
- OpenCL used for multicore CPU, GPU, FPGA (SOpenCL)
- Fast exploration based on area, performance and power requirements

MULTICORE CPU and GPU IMPLEMENTATIONS

- Intel Core i7-4820K CPU clocked at 3.70GHz
  - 8 threads. Each thread performs a random walk
- GeForce GTX 680 GPU (Kepler architecture)
  - Almost 2 Teraflops peak performance, 288 Watts power dissipation
  - N*768 threads (N is the number of points in the grid)
- Use polyhedral analysis to partition original computation in smaller chunks for execution in heterogeneous systems

FPGA IMPLEMENTATIONS

- Good match for FPGA technology:
  - MC algorithm massively parallel
  - Independent multiple path traversal from multiple points
- Poor match for FPGA technology:

–Double precision (DP) Trigonometric, Log, Additions, Multiplications functions for each walk

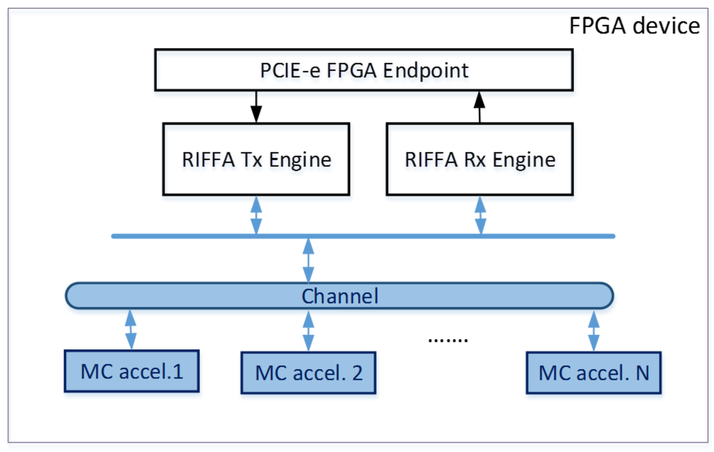
–DP arithmetic takes up a lot of area and is slow

•Use SOpenCL and Xilinx Vivado HLS (High Level Synthesis) tools to automatically generate MC hardware accelerators

•Multiple accelerators with different performance vs. area characteristics can be generated very fast

•Best approach is when one accelerator traverses all walks from as many points as possible

•Target clock 250 MHz

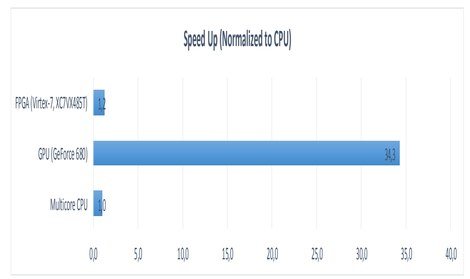


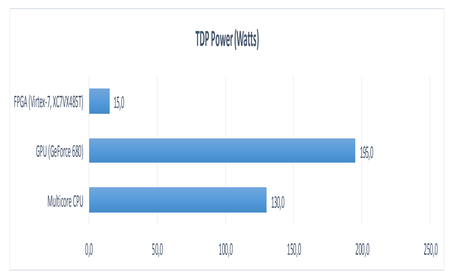
Parameterizable Architecture in Number of Accelerators

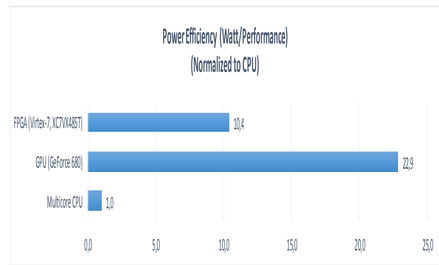
Minimal I/O required : only initial point coordinates.

7. RESULTS

- GPU speed-up is 35X with respect to CPU
- FPGA speed-up is 1.7X with respect to CPU
- VC707 can fit 7 MC accelerators which operate in parallel
- DP arithmetic creates very large FPGA circuits which are slow (~15000 clock cycles per walk).
- However, FPGAs consume less power than both GPU and CPU (TDP: Thermal Design Power)
- GPUs are more power efficient, mainly due to their high performance







8. CONCLUSIONS

- Massively parallel platforms such as GPUs and FPGA can offer higher performance and power efficiency than conventional multicore CPUs for Monte Carlo algorithms
- GPUs are higher performance but higher energy as well.
- CPU platforms can offload computationally expensive parts of the code to GPUs/FPGAs