GPU research in the ES-group

Henk Corporaal (professor)

Gert-Jan van den Braak (postdoc)

Roel Jordans (postdoc)

Erkan Diken (PhD)

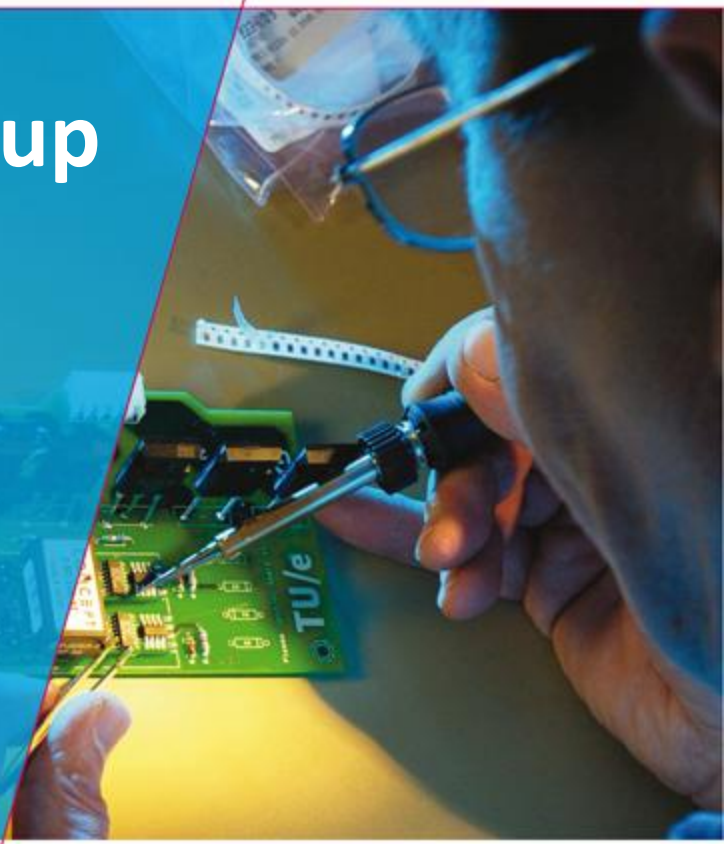
Rik Jongerius (PhD)

Ang Li (PhD)

Maurice Peemen (PhD)

Luc Waeijen (PhD)

Mark Wijtvliet (PhD)



TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

Electronic Systems

PARsE research – http://parse.ele.tue.nl/

Parallel Architecture Research Eindhoven

Log in

- Home
- Research Log
- Research
  - Projects
  - Publications
  - Algorithms and tools
  - Conferences
- Education
  - Courses
  - Student projects
  - GPU cluster project
- Events
  - GPU Mini Symposium
  - GPGPU Symposium
- About us

Parallel Architecture Research Eindhoven - PARsE

Welcome to PARsE

Welcome to the website of the parallel architecture research team. The PARsE team is a subdivision of the Electronic Systems group, part of the Electrical Engineering departement at Eindhoven University of Technology (TU/e) in The Netherlands.

TU/e Technische Universiteit Eindhoven University of Technology

Contents of the website

This website contains information on previous, current and future research, education and events. We update the *research log* daily with new posts concerning the latest updates in research, discussing articles published at journals and conferences from among others ACM and IEEE. You will find the following sections on the website:

- A daily updated **research log**.
- An overview of our **research**, including projects, tools and publications.
- The **education** page, containing course and student project information.
- **Events**, such as our GPU symposium.
- More information **about us**.

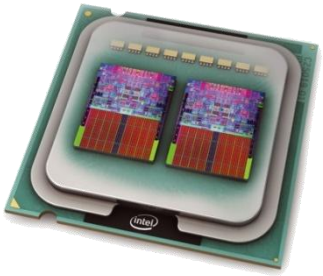
Latest research log entries

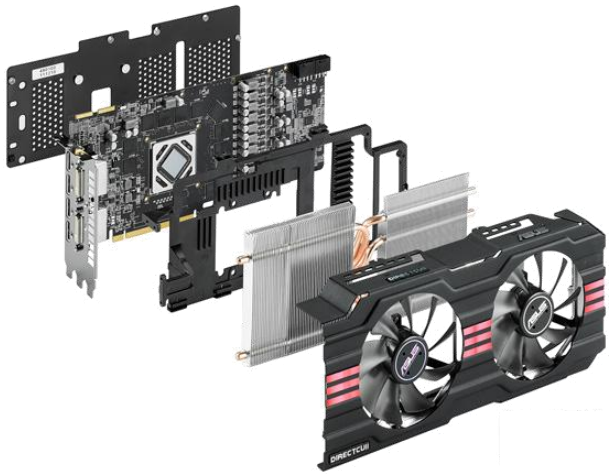
Microserver has landed

The first microserver from IBM/Astron has arrived at the TU/e. We have started benchmarking the microserver. Next, we plan to update our Bones source-to-source compiler and add the microserver as a target in the coming weeks.

PARsE
Parallel Architecture Research Eindhoven

- Using advanced heterogeneous platforms
  - Multi-core CPUs
  - GPUs
  - DSPs
  - FPGAs



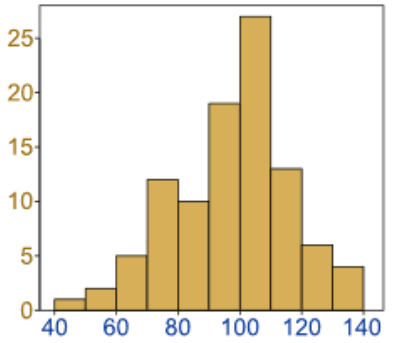


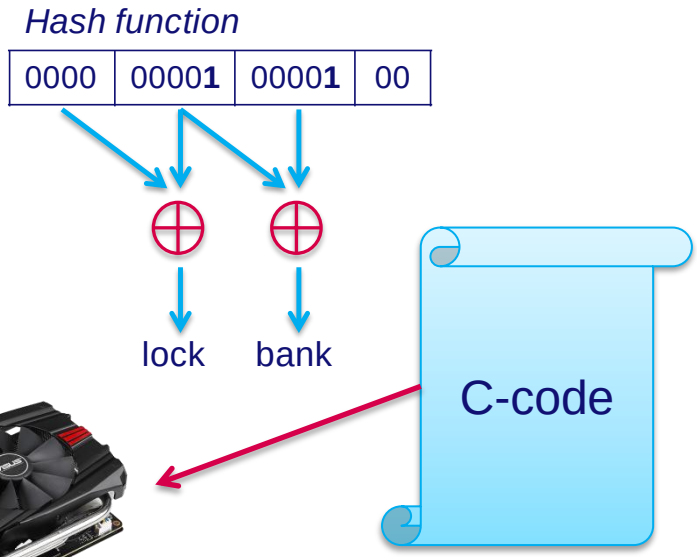
- Efficient code generation
  - Code transformation & generation
  - Compilers
- Even more efficient: new architectures
  - SIMD, CGRA, R-GPU
  - Accelerators
    - Neural networks (CNNs)

Technische Universiteit Eindhoven University of Technology

GPU research – overview (selection)

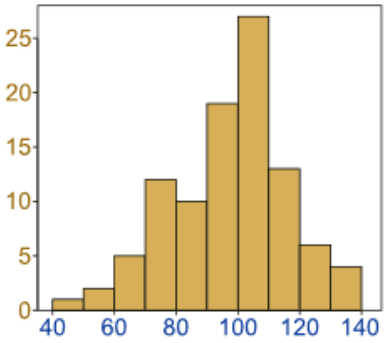
- **Application mapping**
  - **Histogram, CNN**
- **Understanding GPUs**
  - **Modeling of GPU L1 cache**
  - **Cache bypassing**
- **Architecture modification**
  - **Hash functions in scratchpad memory**
- **Code generation**
  - **Bones source-to-source tools**



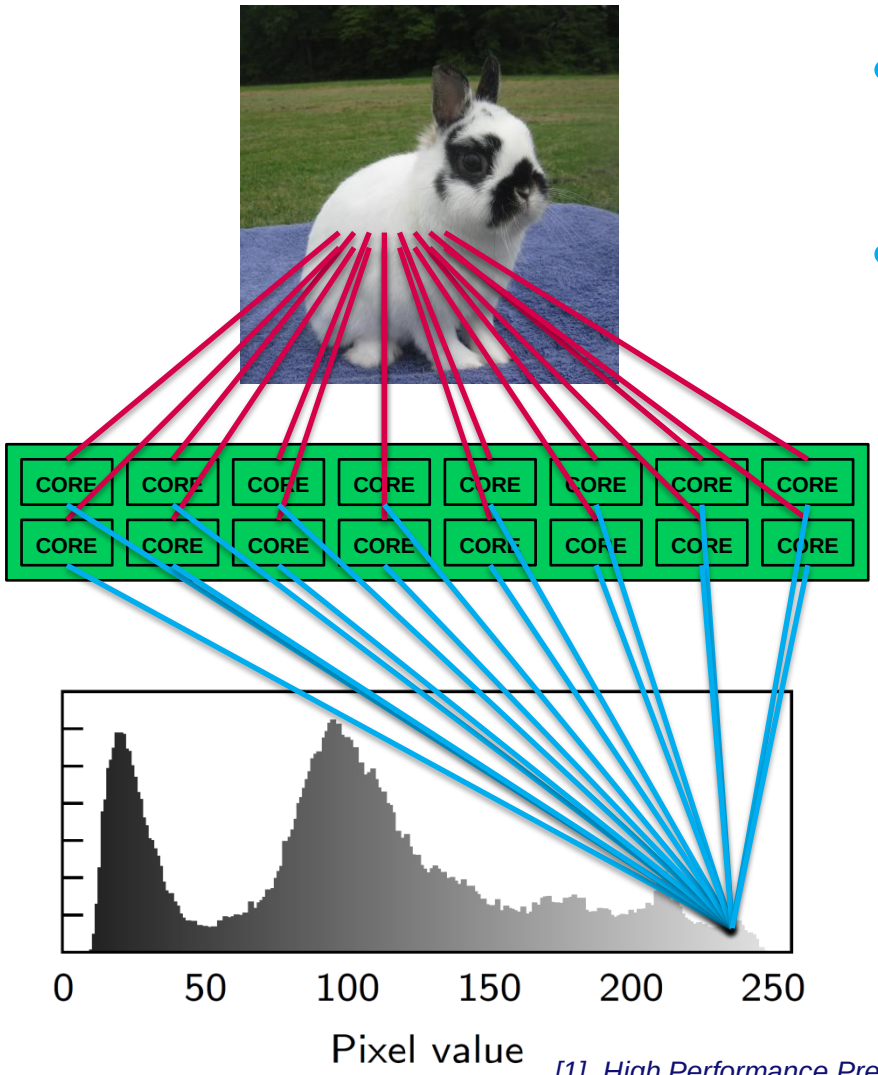


Application mapping

- **Histogram,**
- **Convolutional Neural Networks (CNN)**



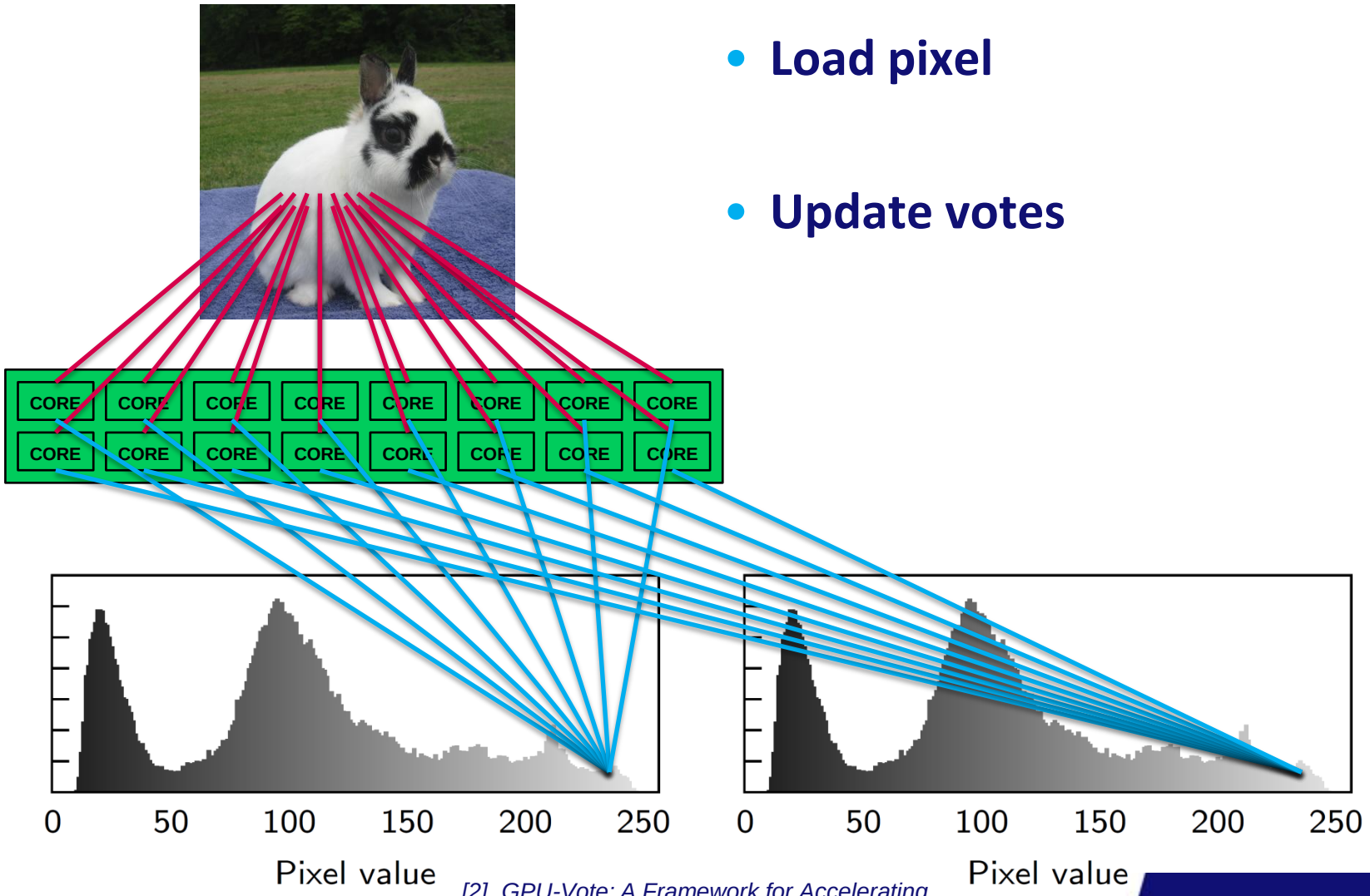
Application mapping: histogram



- **Load pixel**
- **Update votes**

[1] *High Performance Predictable Histogramming on GPUs: Exploring and Evaluating Algorithm Trade-offs*

Histogram – replication

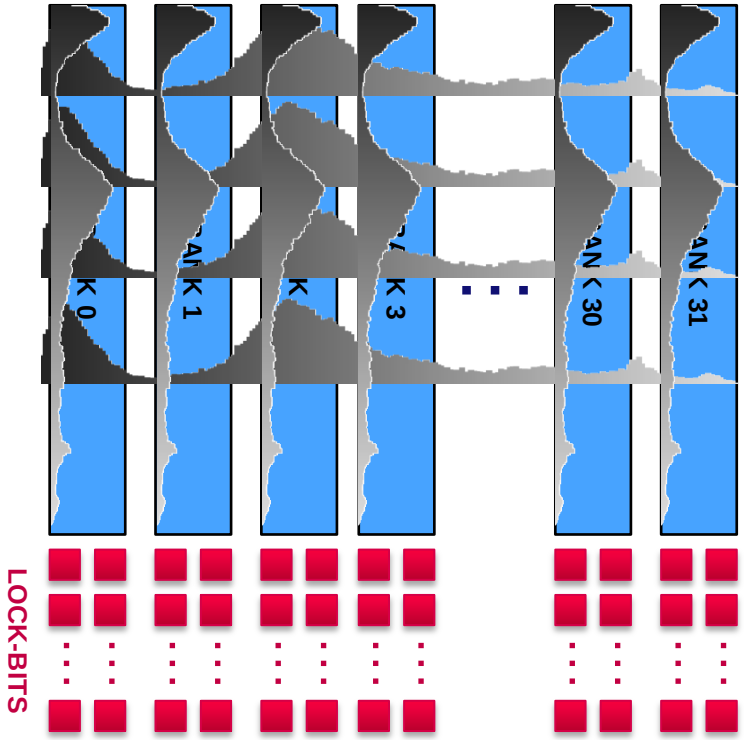


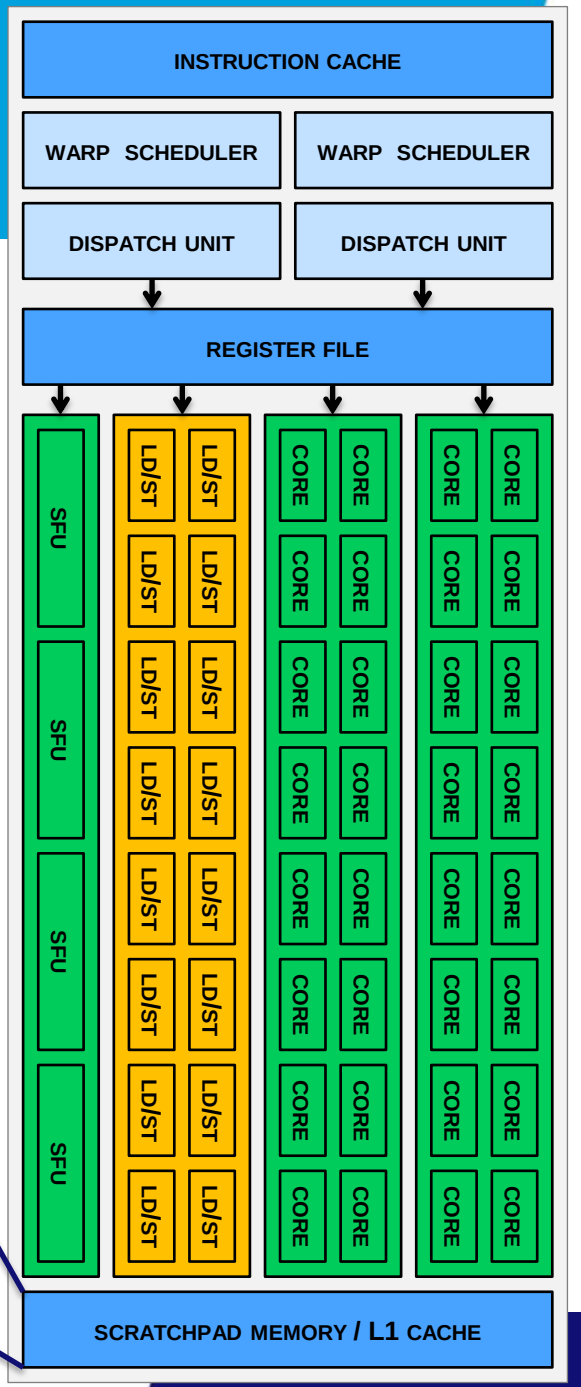
- **Load pixel**
- **Update votes**

[2] *GPU-Vote: A Framework for Accelerating Voting Algorithms on GPU*

Scratchpad memory layout

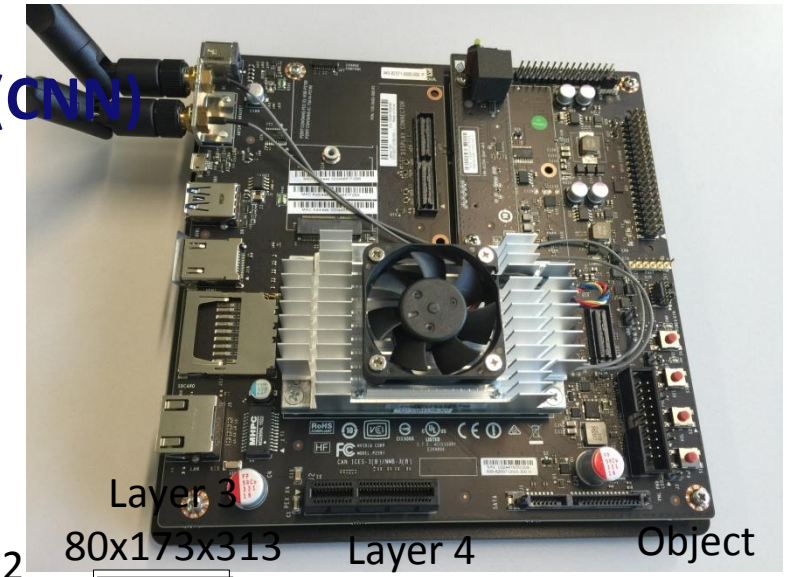
- Scratchpad memory
  - Divided in 32 banks
  - Each bank has 32 lock-bits, 1024 in total

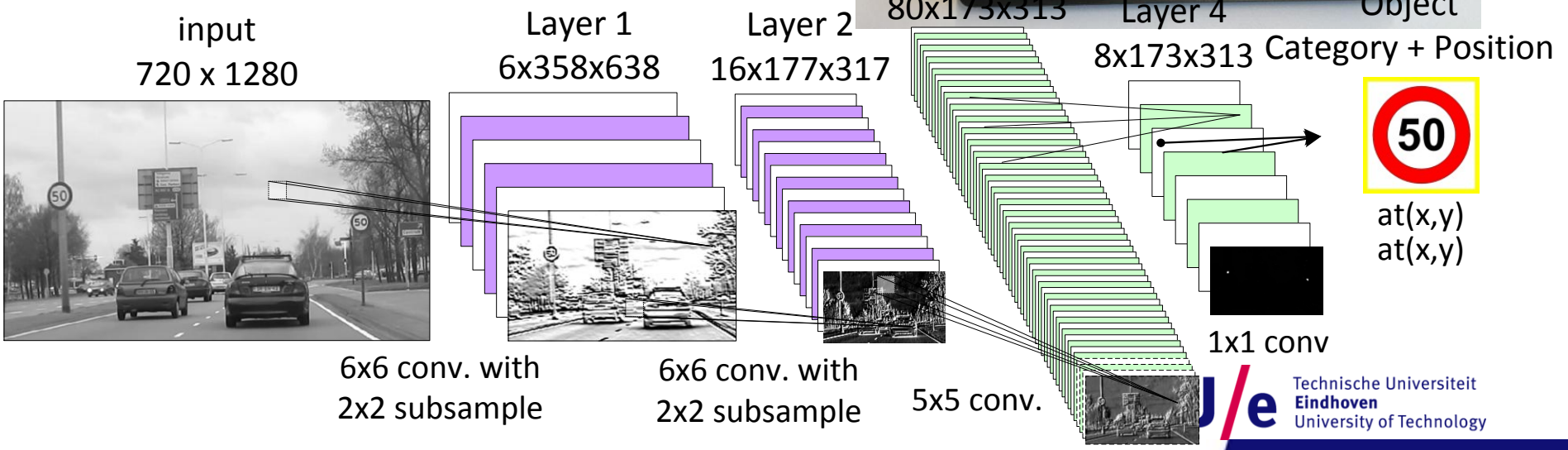




Application mapping: CNN

- **Convolutional Neural Network (CNN)**
  - **GTX 460: 35fps**
  - **Tegra X1: ~20fps**

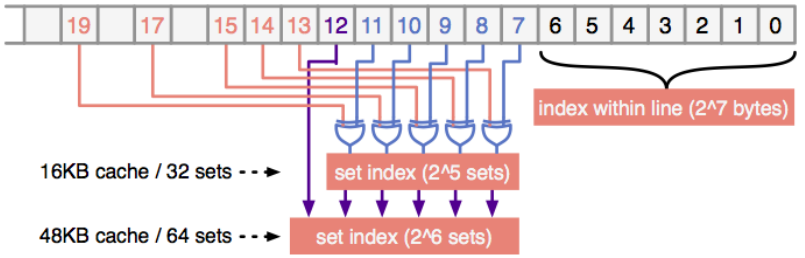




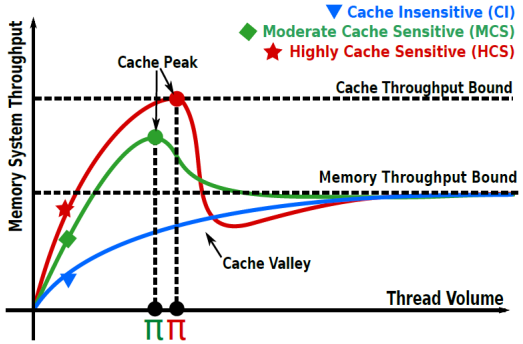
[4] Speed Sign Detection and Recognition by Convolutional Neural Networks

Understanding GPUs

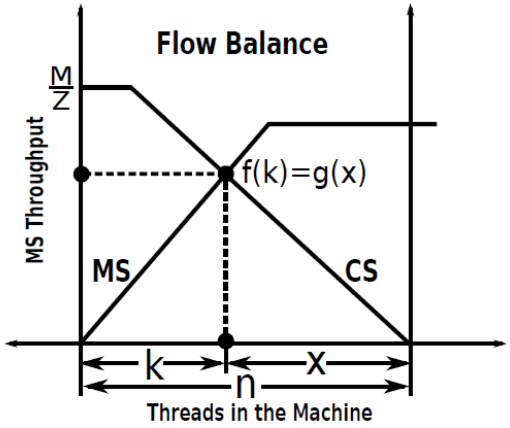
- **Modeling of GPU L1 cache**



- **Cache bypassing**

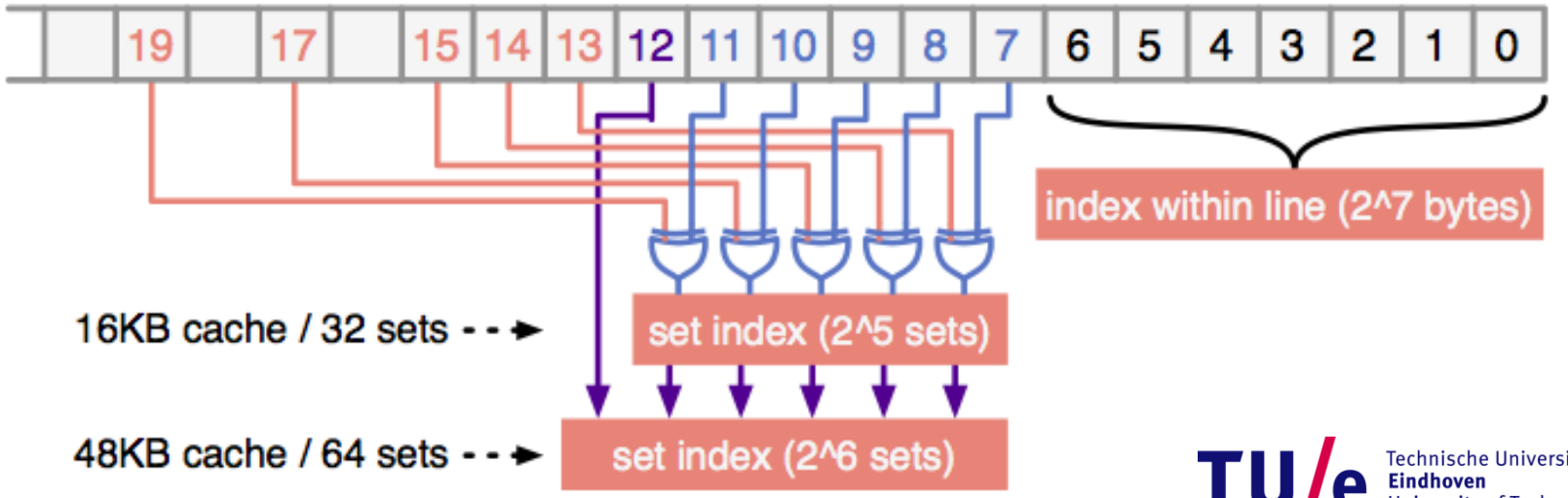


- **Transit model**



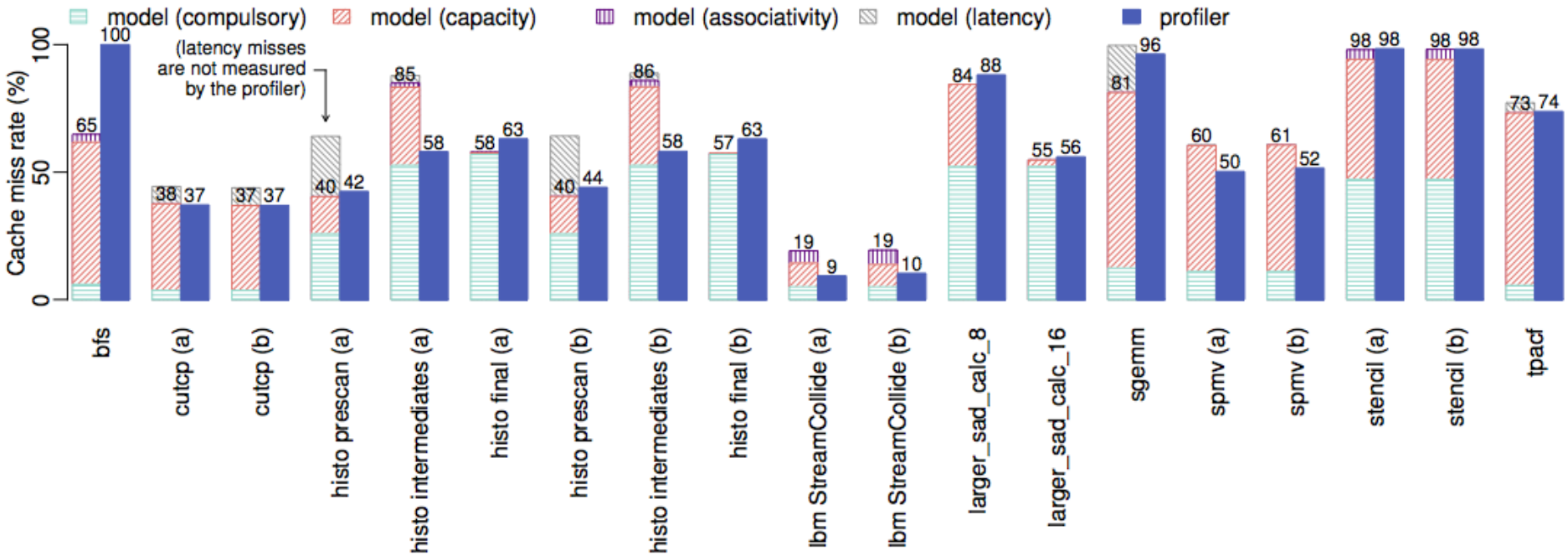
Understanding GPUs: L1 cache modeling

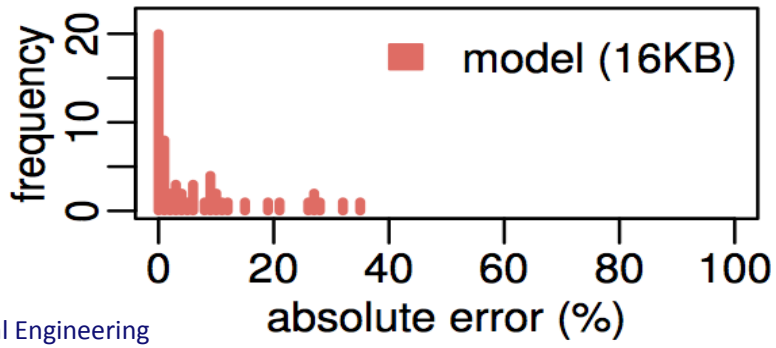
- **GPU Cache model:**
  - **Execution model (threads, thread blocks)**
  - **Memory latencies**
  - **MSHRs (pending memory requests)**
  - **Cache associativity**



[5] A Detailed GPU Cache Model Based on Reuse Distance Theory

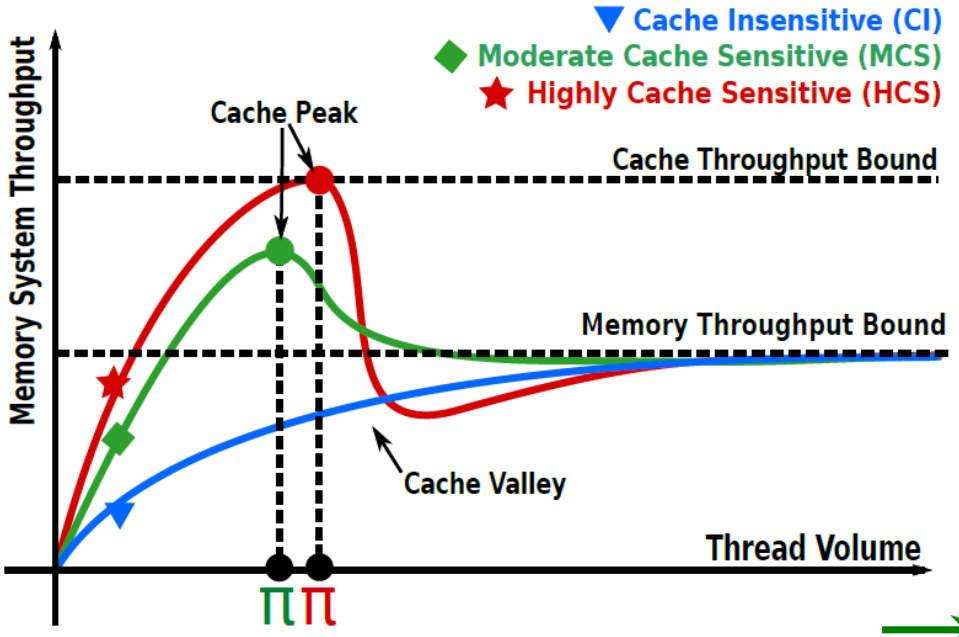
L1 cache model – results

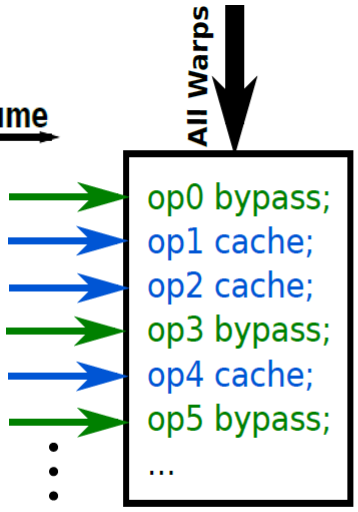




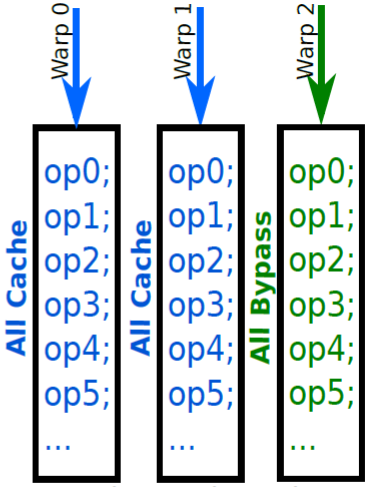
Mean absolute error of 6.4%

Understanding GPUs: Cache bypassing





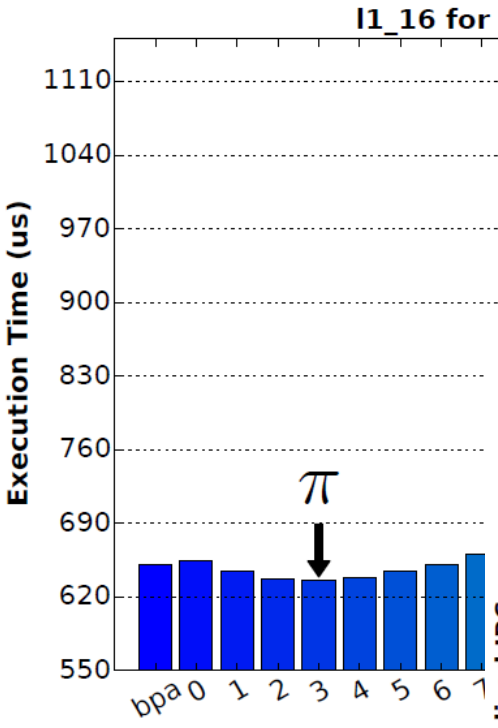
Vertical Design
(Bypass based on Operations)

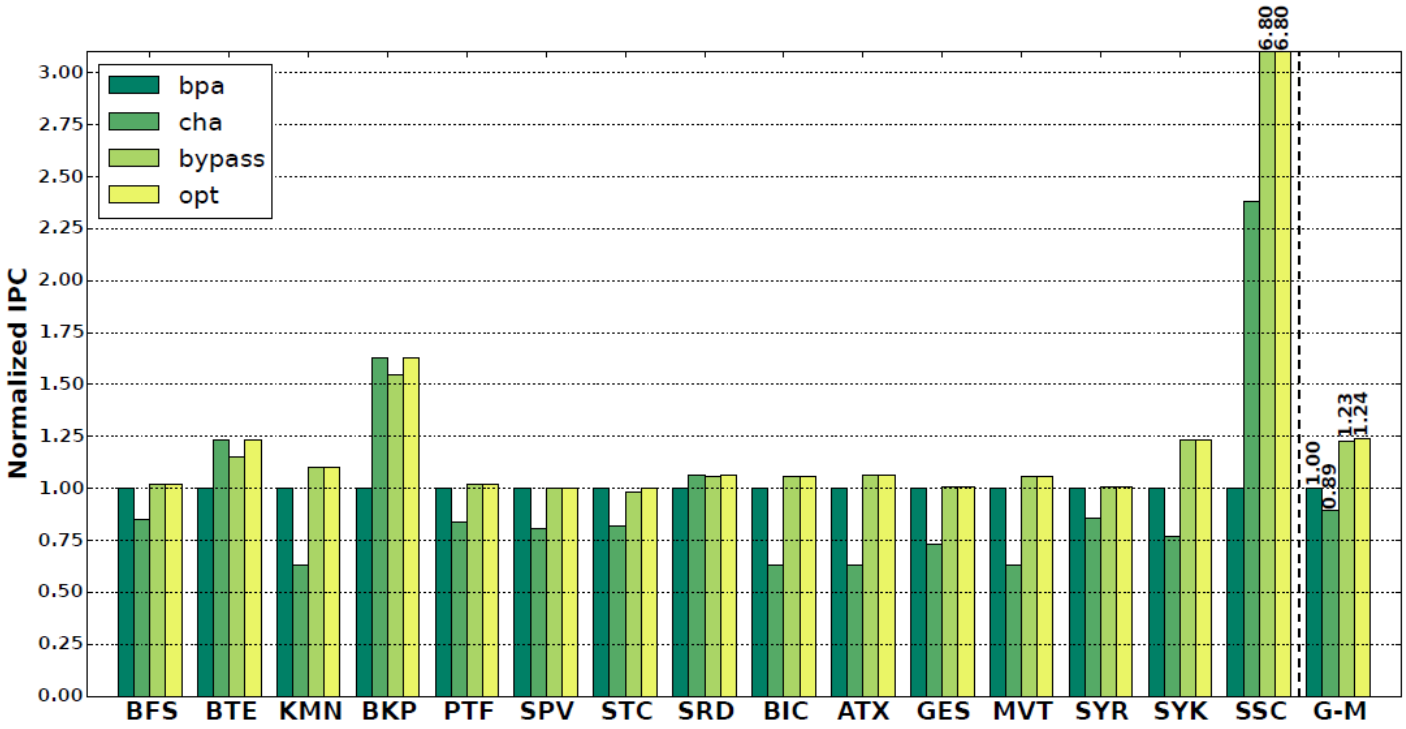


Horizontal Design
(Bypass based on Warp Index)

[6] *Adaptive and Transparent Cache Bypassing for GPUs*

Cache bypassing – results

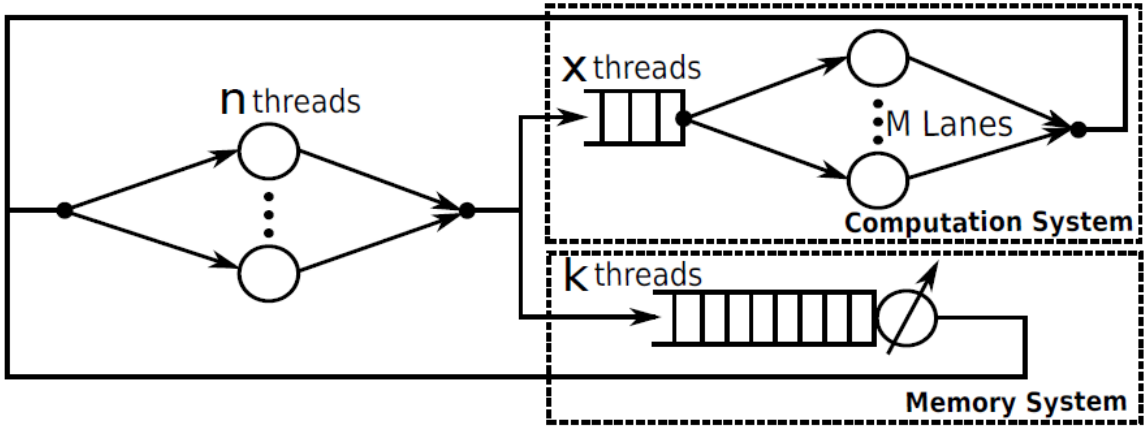


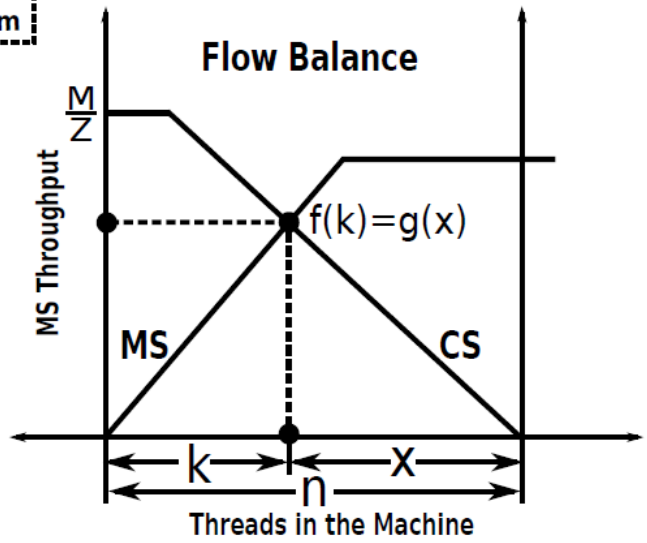


[6] Adaptive and Transparent Cache Bypassing for GPUs

Understanding GPUs: Transit model

- **Transit model: computation and memory sub-systems**

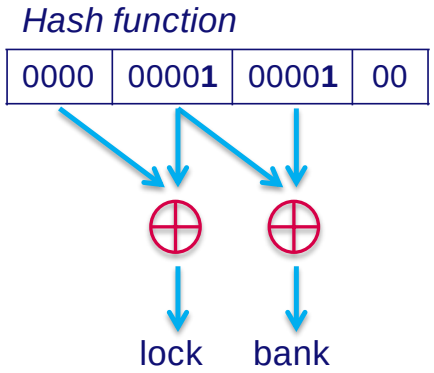




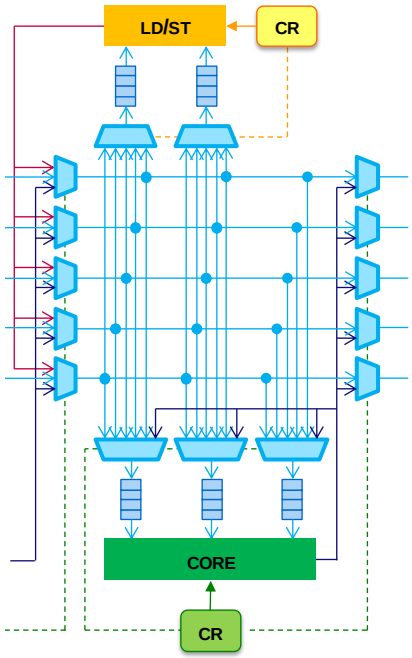
[7] Transit: A Visual Analytical Model for Multithreaded Machines

Architecture modifications

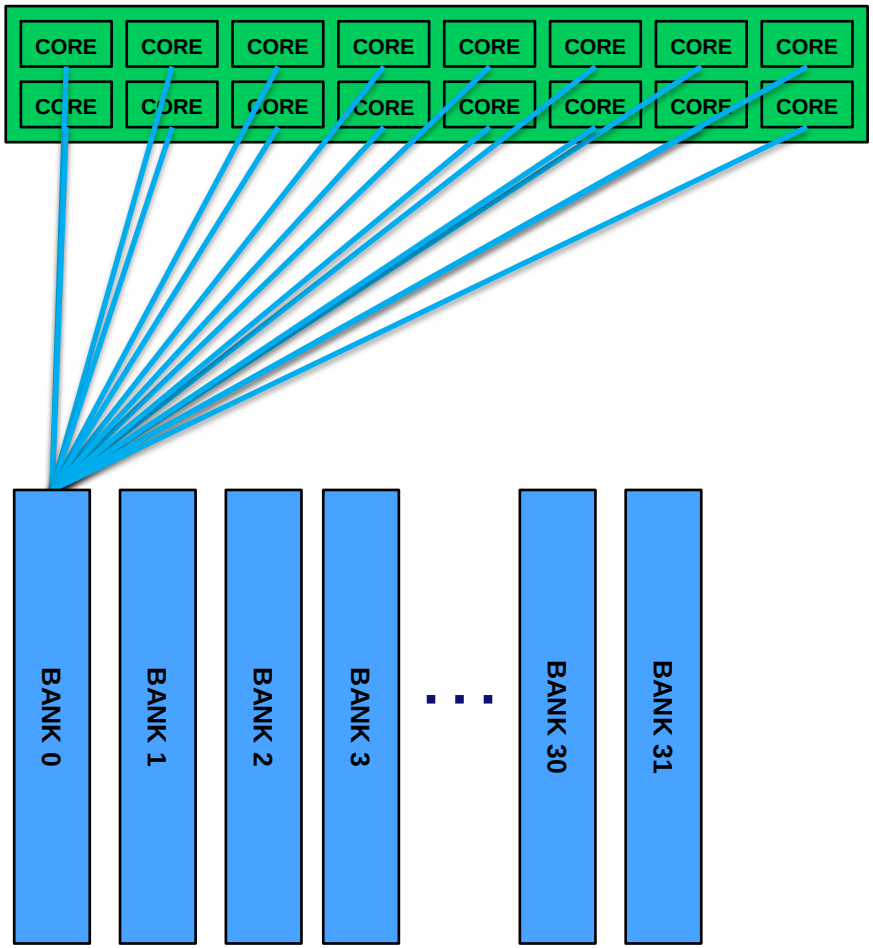
- **Scratchpad memory hash functions**



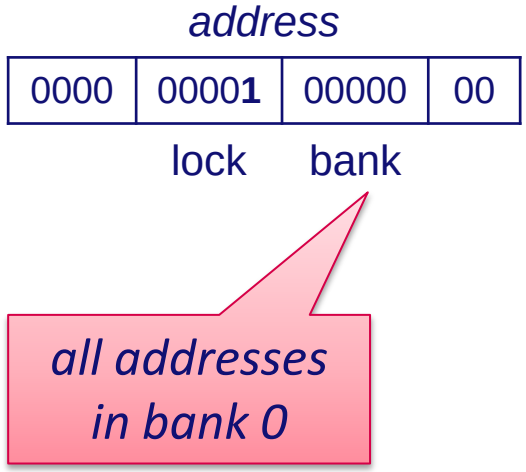
- **R-GPU**



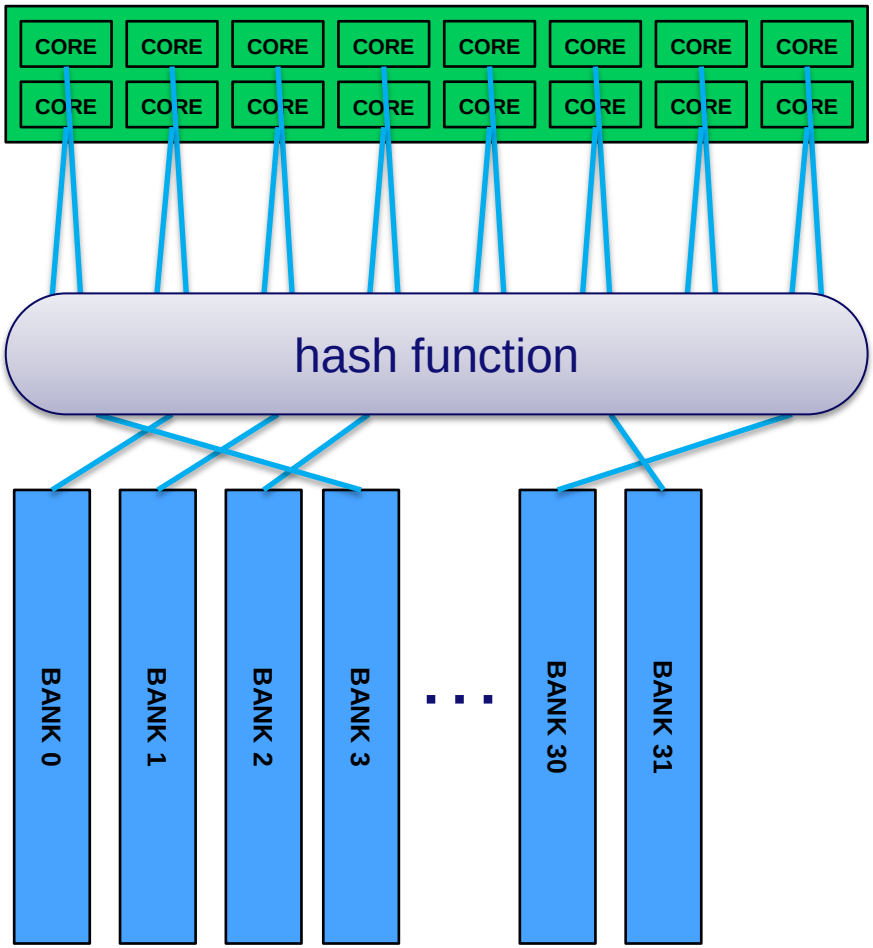
GPU modifications: bank & lock conflicts



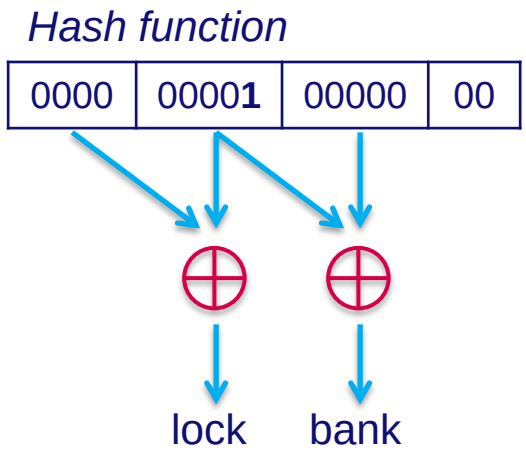

```
addr = 32 * id
```



Resolving bank conflicts: hash functions

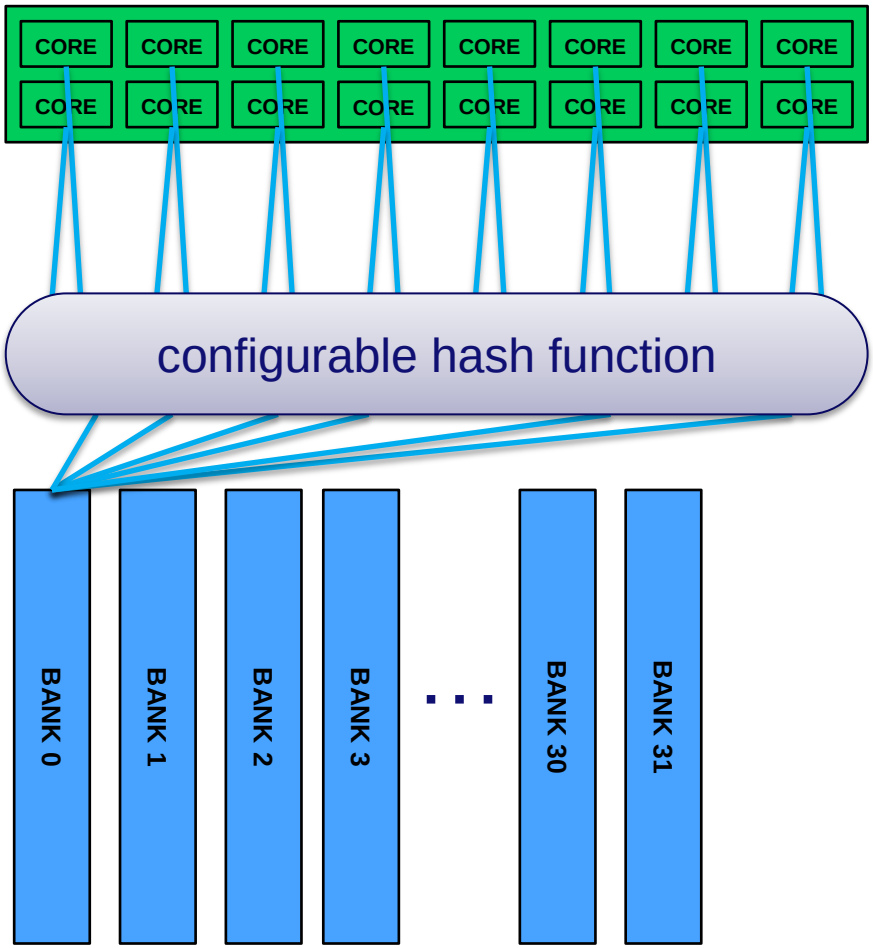



```
addr = 32 * id
```

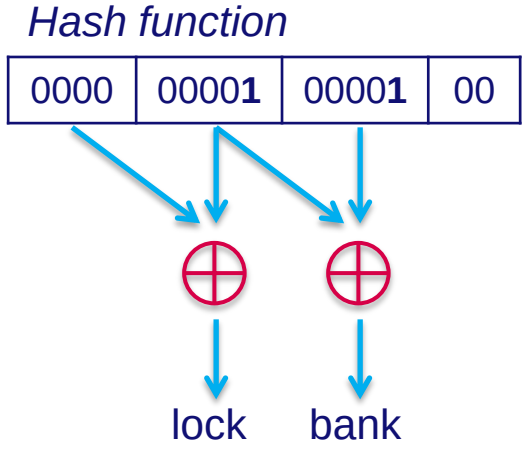


[8] *Simulation and Architecture Improvements of Atomic Operations on GPU Scratchpad Memory*

Resolving bank conflicts: hash functions

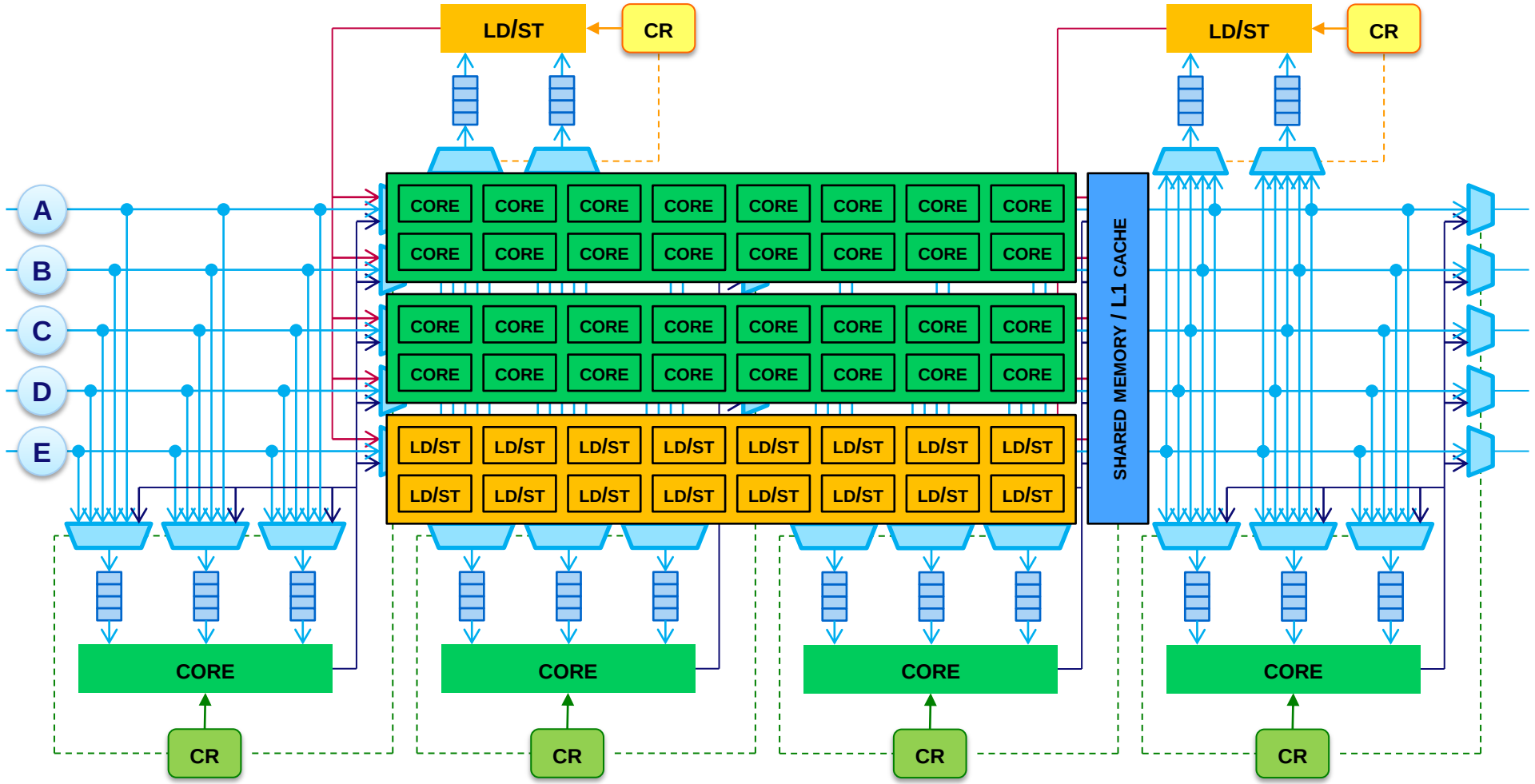



```
addr = 33 * id
```

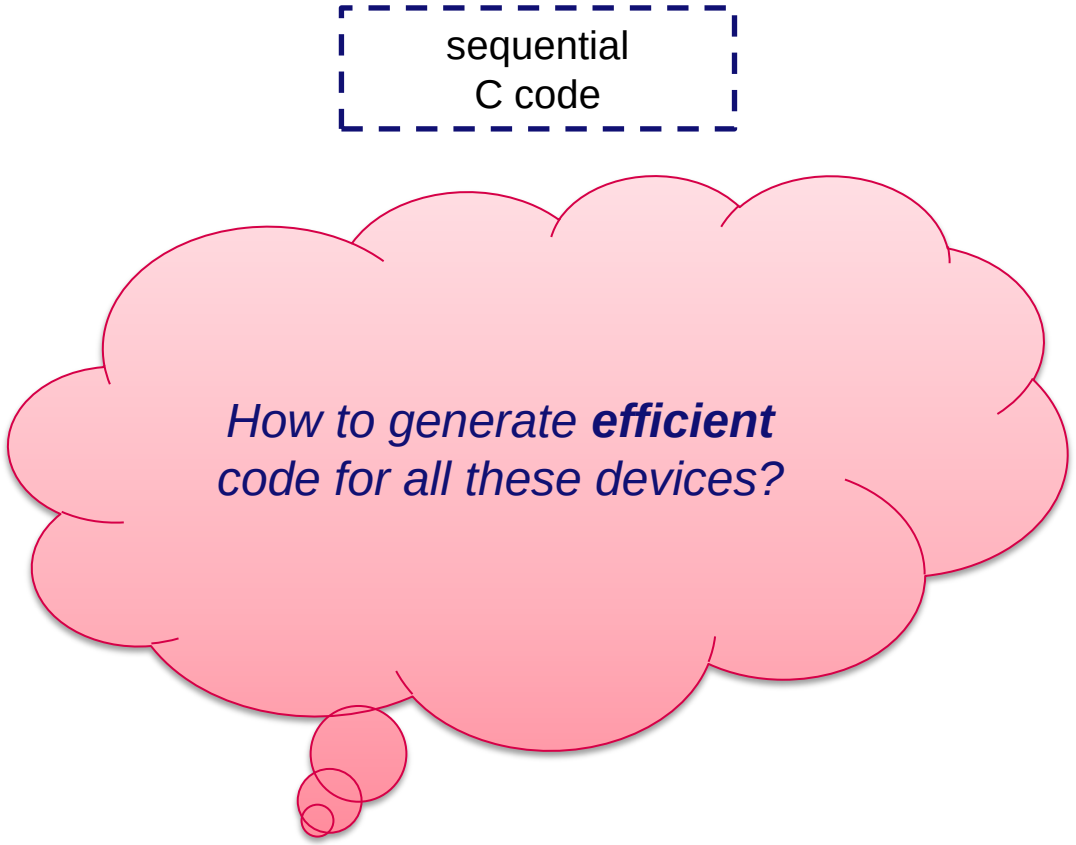


[9] *Configurable XOR Hash Functions for Banked Scratchpad Memories in GPUs*

Architecture modifications: R-GPU

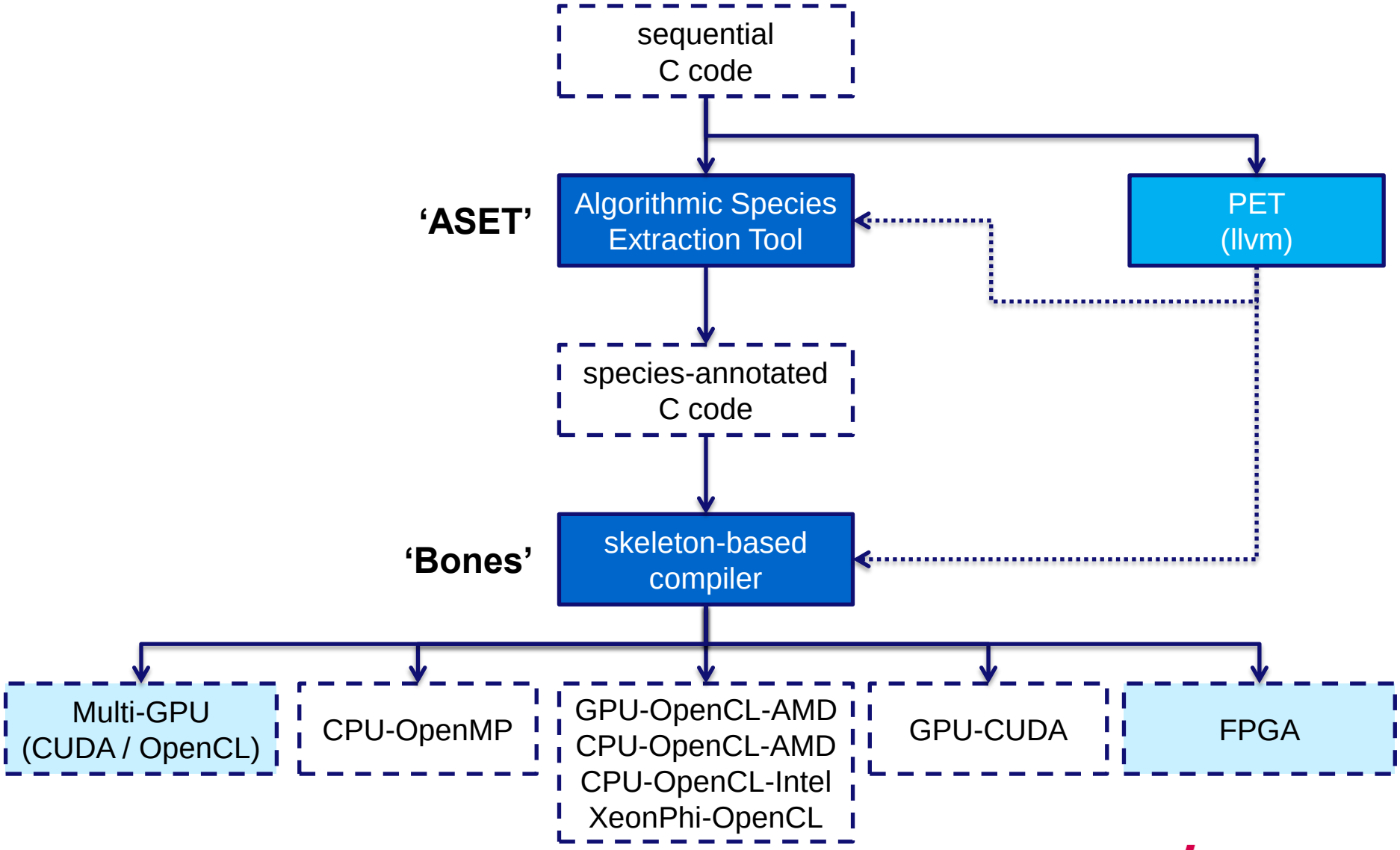


Code generation: ASET & Bones



[10] Automatic Skeleton-Based Compilation through Integration with an Algorithm Classification

Code generation: ASET & Bones

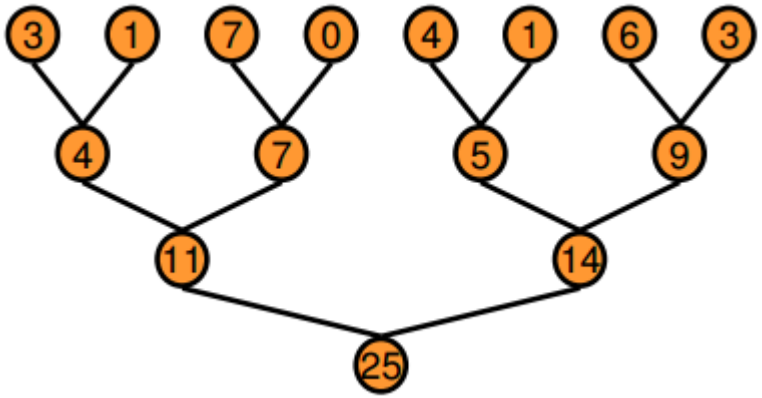


[10] Automatic Skeleton-Based Compilation through Integration with an Algorithm Classification

Example C to CUDA transformation

Example 1: Sum

```
int sum = 0;
for (int i=0; i<N; i++){
  sum = sum + in[i];
}
```




```
template <unsigned int blockSize>
__device__ void warpReduce(volatile int *sm, unsigned int tid) {
if (blockSize >= 64) sm[tid] += sm[tid + 32];
if (blockSize >= 32) sm[tid] += sm[tid + 16];
if (blockSize >= 16) sm[tid] += sm[tid +  8];
if (blockSize >=  8) sm[tid] += sm[tid +  4];
if (blockSize >=  4) sm[tid] += sm[tid +  2];
if (blockSize >=  2) sm[tid] += sm[tid +  1];
}

template <unsigned int blockSize>
__global__ void reduce6(int *g_idata, int *g_odata, unsigned int n) {
   extern __shared__ int sm[];
   unsigned int tid = threadIdx.x;
   unsigned int i = blockIdx.x*(blockSize*2) + tid;
   unsigned int gridSize = blockSize*2*gridDim.x;
   sm[tid] = 0;
   while (i < n) {
      sm[tid] += g_idata[i]
      sm[tid] += g_idata[i+blockSize];
      i += gridSize;
   }
   __syncthreads();
   if (blockSize >= 512) {
      if (tid < 256) { sm[tid] += sm[tid + 256]; }
      __syncthreads();
   }
   if (blockSize >= 256) {
      if (tid < 128) { sm[tid] += sm[tid + 128]; }
      __syncthreads();
   }
   if (blockSize >= 128) {
      if (tid <  64) { sm[tid] += sm[tid +  64]; }
      __syncthreads();
   }
   if (tid < 32) { warpReduce<blockSize>(sm, tid); }
   if (tid == 0) { g_odata[blockIdx.x] = sm[0]; }
}
```

Second example: maximum

3 1 7 0 4 1 6 3

Example 1: Sum

```
int sum = 0;
for (int i=0; i<N; i++){
  sum = sum + in[i];
}
```

Example 2: Max

```
int max = 0;
for (int i=0; i<N; i++){
  max = (max>in[i])?max:in[i];
}
```

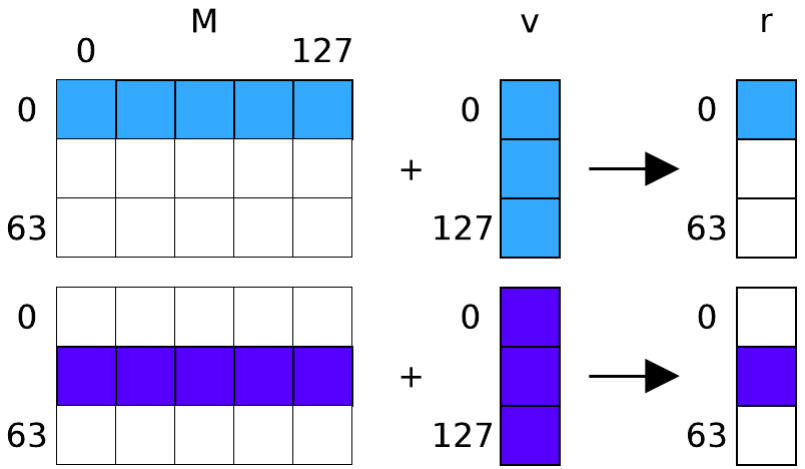
```
template <unsigned int blockSize>
__device__ void warpReduce(volatile int *sm, unsigned int tid) {
if (blockSize >= 64) sm[tid] = (sm[tid]>sm[tid+32]) ? sm[tid] : sm[tid+32];
if (blockSize >= 32) sm[tid] = (sm[tid]>sm[tid+16]) ? sm[tid] : sm[tid+16];
if (blockSize >= 16) sm[tid] = (sm[tid]>sm[tid+ 8]) ? sm[tid] : sm[tid+ 8];
if (blockSize >=  8) sm[tid] = (sm[tid]>sm[tid+ 4]) ? sm[tid] : sm[tid+ 4];
if (blockSize >=  4) sm[tid] = (sm[tid]>sm[tid+ 2]) ? sm[tid] : sm[tid+ 2];
if (blockSize >=  2) sm[tid] = (sm[tid]>sm[tid+ 1]) ? sm[tid] : sm[tid+ 1];
}

template <unsigned int blockSize>
__global__ void reduce6(int *g_idata, int *g_odata, unsigned int n) {
   extern __shared__ int sm[];
   unsigned int tid = threadIdx.x;
   unsigned int i = blockIdx.x*(blockSize*2) + tid;
   unsigned int gridSize = blockSize*2*gridDim.x;
   sm[tid] = 0;
   while (i < n) {
      sm[tid] = (sm[tid]>g_idata[i]) ? sm[tid] : g_idata[i];
      sm[tid] = (sm[tid]>g_idata[i+blockSize]) ? sm[tid] : g_idata[i+blockSize];
      i += gridSize;
   }
   __syncthreads();
   if (blockSize >= 512) {
      if (tid < 256) { sm[tid] = (sm[tid]>sm[tid+256]) ? sm[tid] : sm[tid+256]; }
      __syncthreads();
   }
   if (blockSize >= 256) {
      if (tid < 128) { sm[tid] = (sm[tid]>sm[tid+128]) ? sm[tid] : sm[tid+128]; }
      __syncthreads();
   }
   if (blockSize >= 128) {
      if (tid <  64) { sm[tid] = (sm[tid]>sm[tid+ 64]) ? sm[tid] : sm[tid+ 64]; }
      __syncthreads();
   }
   if (tid < 32) { warpReduce<blockSize>(sm, tid); }
   if (tid == 0) { g_odata[blockIdx.x] = sm[0]; }
}
```

Algorithmic species

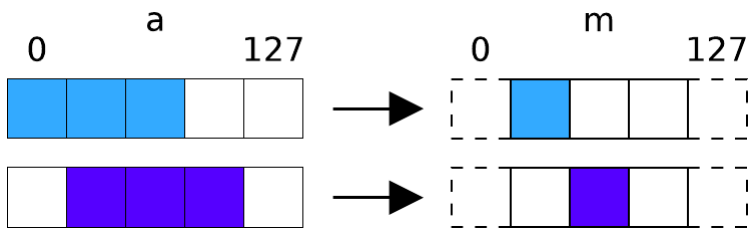
- **Matrix-vector multiplication:**

```
0:63,0:127|chunk(0:0,0:127) ^ 0:127|full → 0:63|element
for (i=0; i<64; i++) {
   r[i] = 0;
   for (j=0; j<128; j++) {
         r[i] += M[i][j] * v[j];
   }
}
```

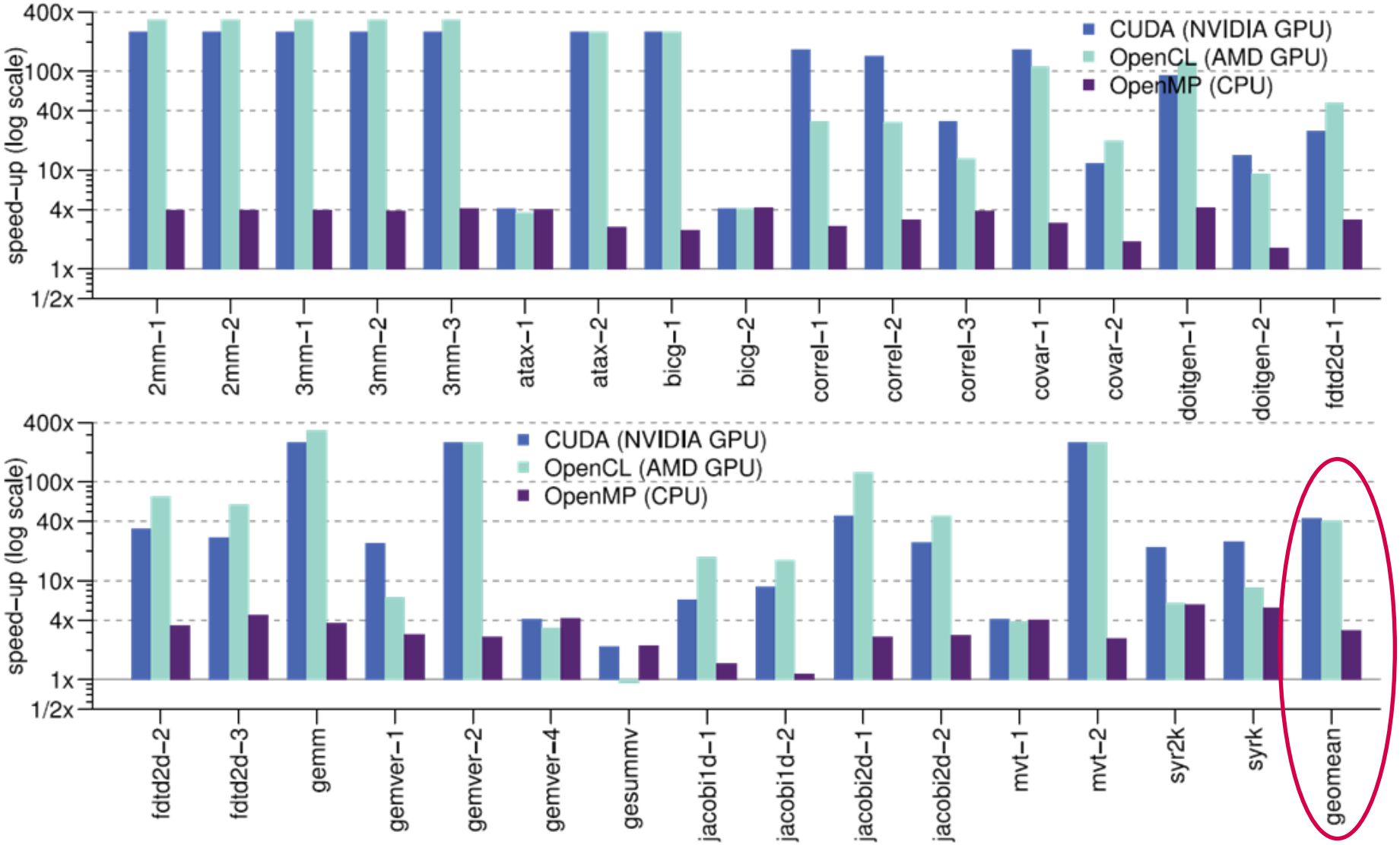


- **Stencil computation:**

```
1:126|neighbourhood(-1:1) → 1:126|element
for (i=1; i<128-1; i++) {
   m[i] = 0.33 * (a[i-1] + a[i] + a[i+1]);
}
```



What do we gain in performance?



[12] Bones: An Automatic Skeleton-Based C-to-CUDA Compiler for GPUs

Education – mapping assignments

- **Master course**
  - **contrast enhancement**
  - **Viola-Jones Face Detection**
  - **SIFT object recognition**
  - **convolutional neural network (CNN)**
  - **‘bitcoin’ mining**

- **Post-master (PDEng) course:**
  - **GPU & cluster computing**

Student projects

- Accelerating AURORA on Multi-Core and Many-Core Processor Architectures – VITO, Belgium

- Advanced ultrasound beam forming using GPGPU technology – esaote, Maastricht

- Domain Transform Acceleration for the GPU-Based Real-Time Planar Near-Field Acoustic Holography

- Analysis and Modeling of the Timing Behavior of GPU Architectures, TU/e

TU/e Technische Universiteit Eindhoven University of Technology

Summary

- Research topics:
  - Application mapping
  - Understanding GPUs
  - Architecture modifications
  - Code generations
- MSc. students, PDEngs & PhDs
- More on the website:
  - http://parse.ele.tue.nl/

PHILIPS

THALES

océ

TU/e Technische Universiteit Eindhoven University of Technology