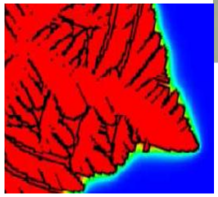
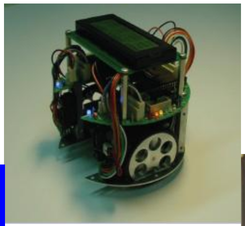
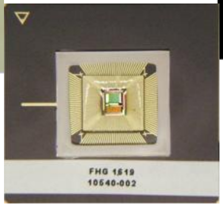
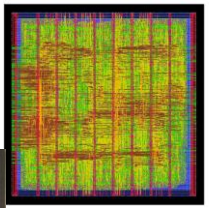
Using Hardware Accelerators for HPC Tasks



Dietmar Fey
Friedrich-Alexander-University Erlangen-Nuremberg
Department Computer Science – Chair for Computer Architecture

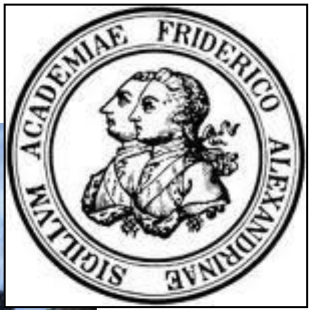
Friedrich-Alexander-Universität Erlangen-Nürnberg

TECHNISCHE FAKULTÄT

The Friedrich-Alexander University Erlangen-Nuremberg

Some data about our university

- founded 1743



- 28.677 students
  - 2.537 are from abroad (8.9%)
- 21.8 % students at the Technical Faculty
  - ~ 20% female

Hardware accelerator architectures for HPC

Hardware accelerators for HPC

- GPU
- FPGA

Examples

- Solving HPC tasks for optical 3D metrology on GPUs
- Solving Lattice-Boltzmann and heat equation kernels on FPGAs

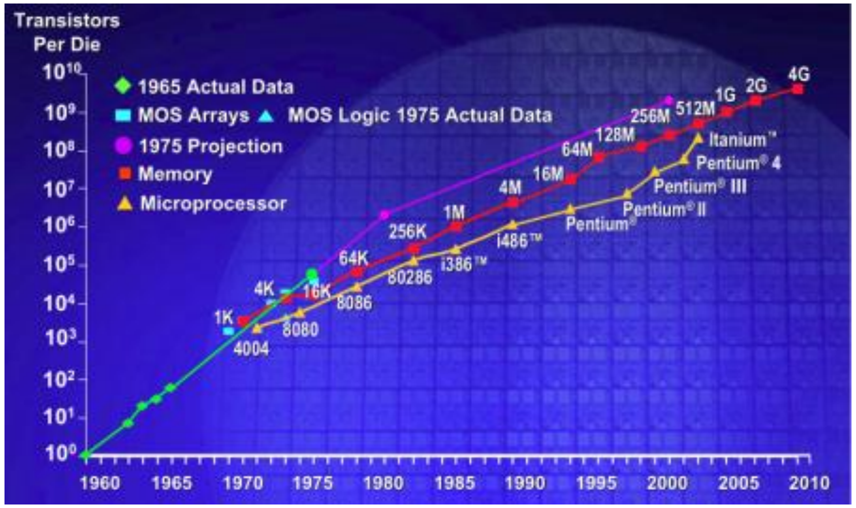
Lecture in the Master Program "Computational Engineering" in Erlangen from Computer Architecture side

- Architecture of Supercomputers

Hardware accelerator architectures for HPC

Why should hardware accelerators be used?

- Is the normal CPU not fast enough?
- No talk in Computer Engineering without Moore’s Law
  - Every 18 to 24 months doubling of devices on chips
  - Keeping pace with Moore‘s law is becoming more and more difficult
  - Slowing down the catching up of the CPU versus accelerator

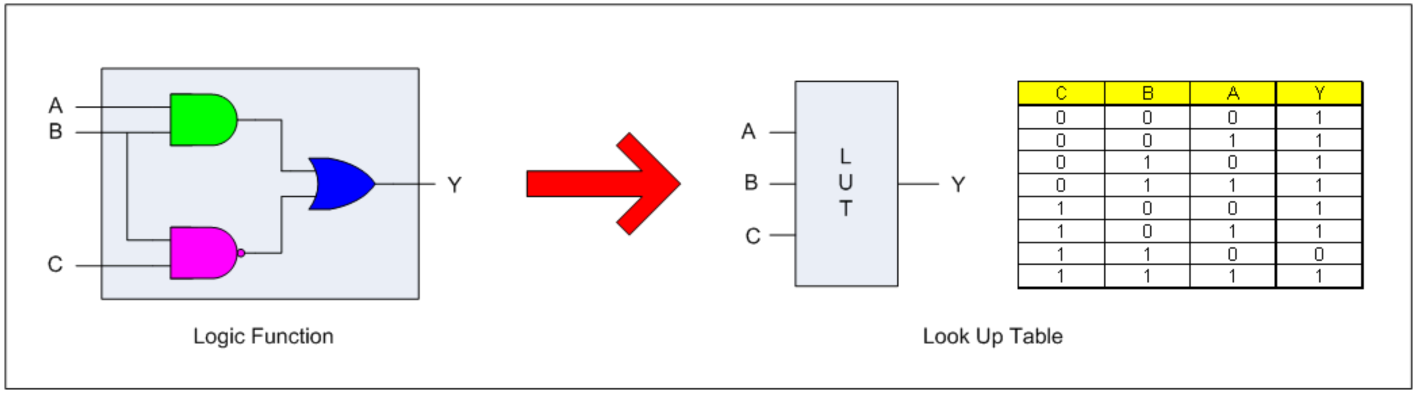


Hardware accelerator architectures for HPC

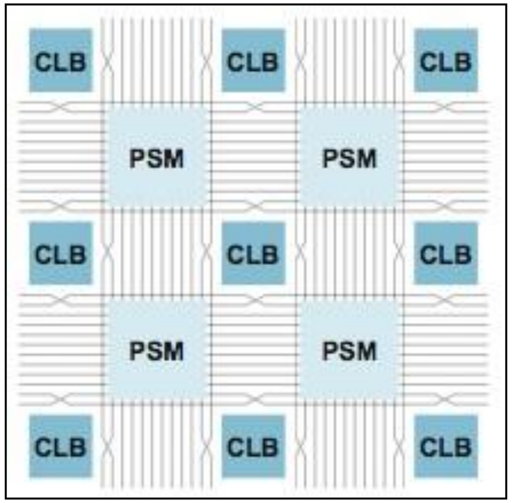
- Which kind of hardware accelerator are considered in the following?
  - GPUs and FPGAs

  - GPU – Graphics Processing Unit
    - used as GPGPU – General Purpose Graphics Processing Unit

Generic (simplified) setup of an FPGA

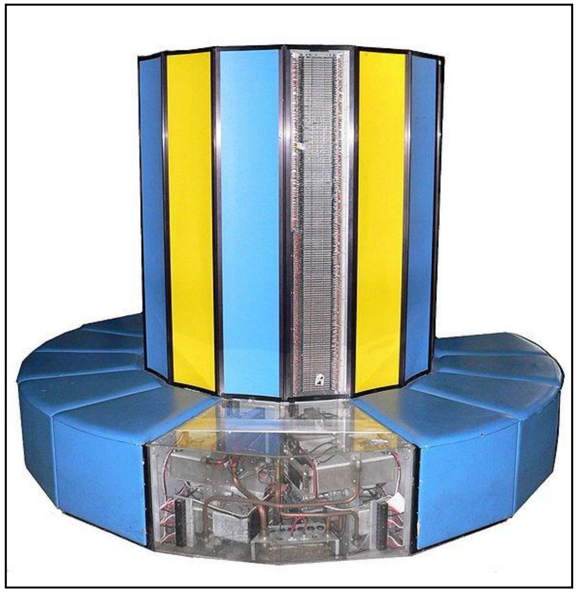


C	B	A	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

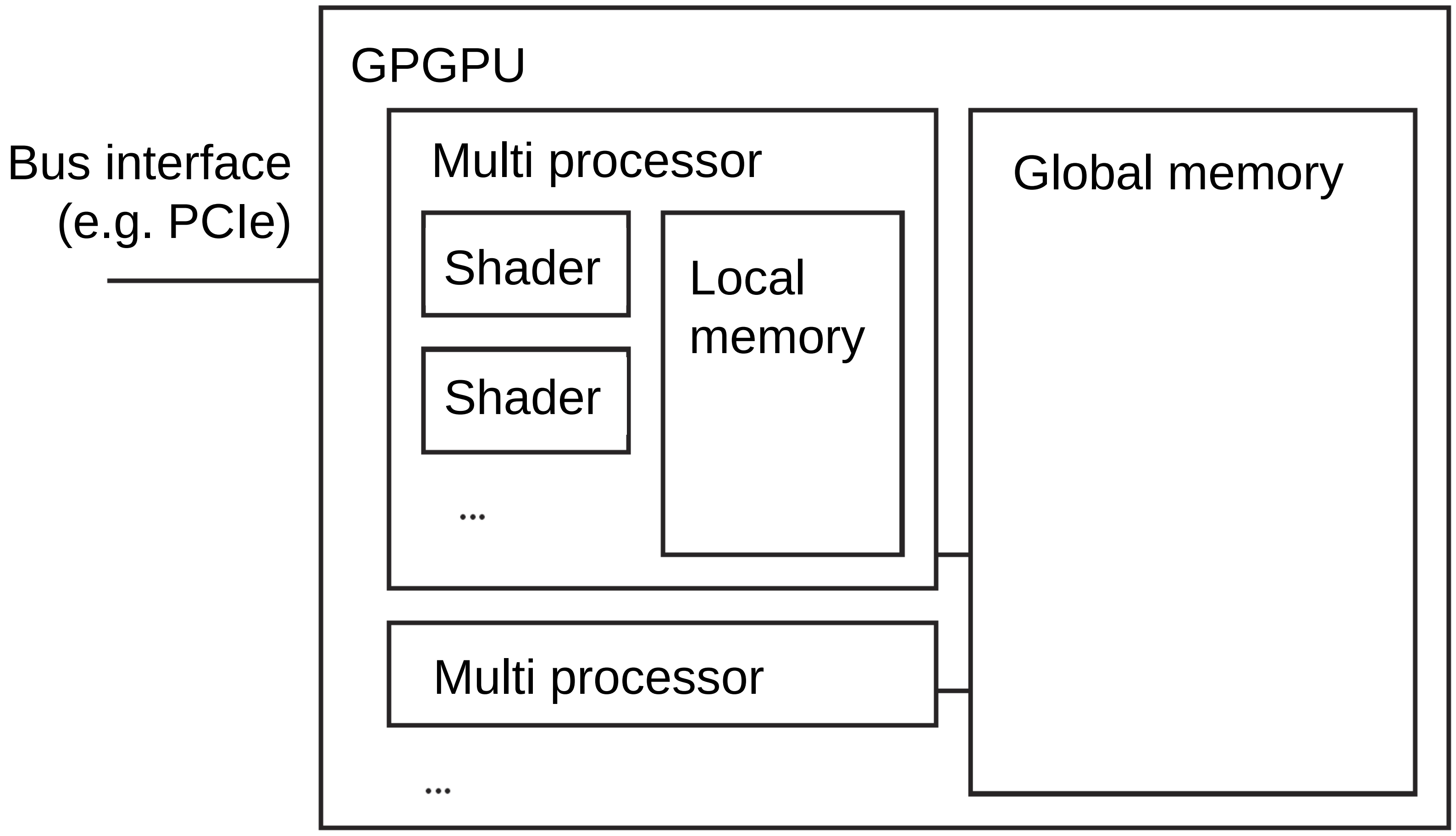


Hardware accelerator architectures for HPC

- FPGA – Field Programmable Gate Arrays
  - Have already been used in Cray Supercomputers, Cray XD1
  - Processing node contained
    - four AMD Opteron 64-bit CPUs
    - and one Xilinx Virtex II – device
  - FPGAs augment the processing or input/output capabilities of the Opteron processors
    - FFT (Fast Fourier Transform)
    - Replacing more slowly PCI interfaces
  - Furthermore, each FPGA contained a pair of PowerPC 405 processors



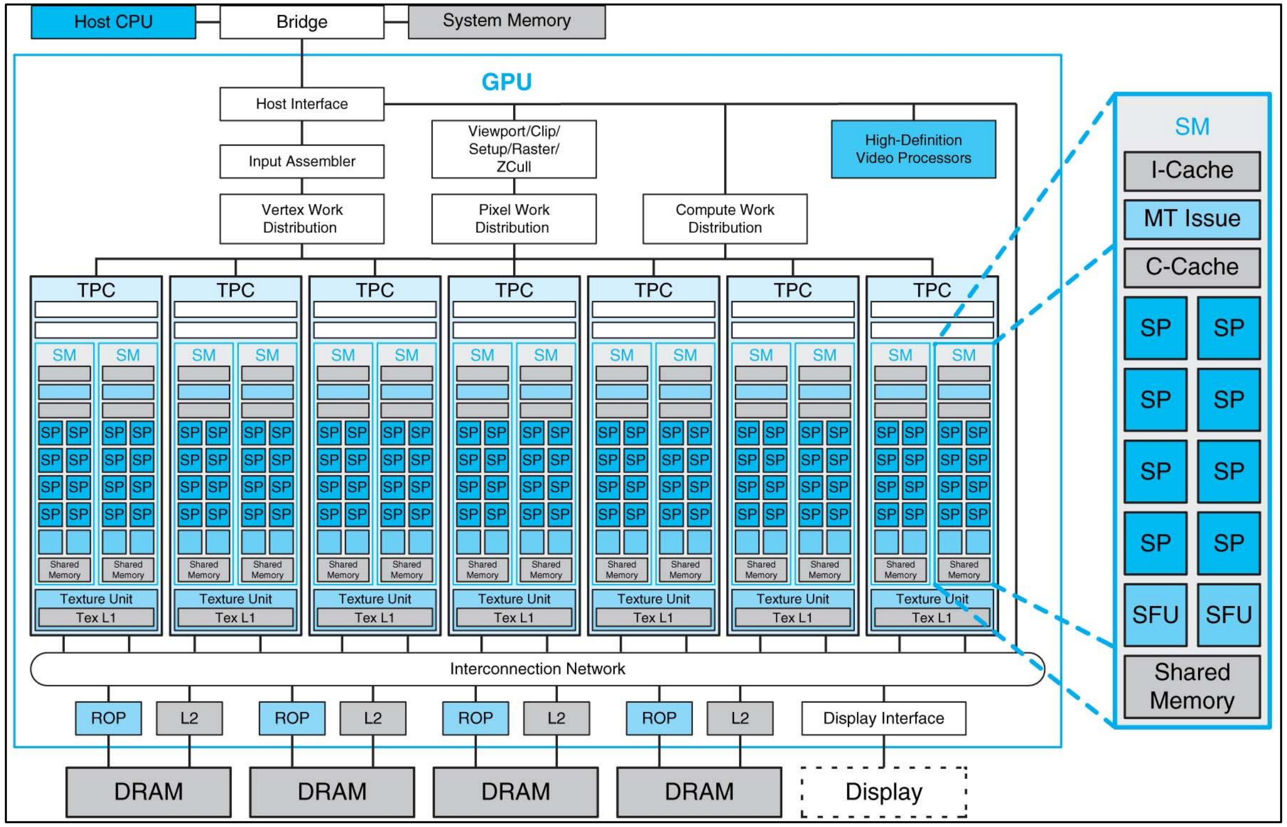
Generic setup of a GPU



Hardware accelerator architectures for HPC

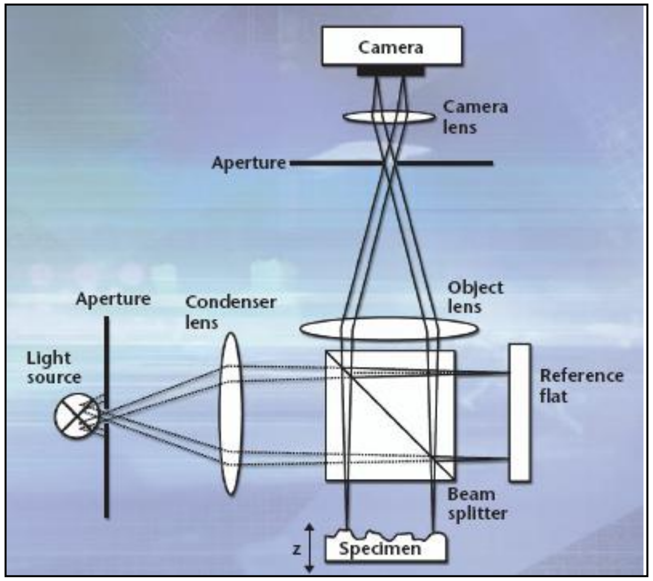
Set-up of a real GPU NVIDIA GeForce 8800

- 112 Streaming processors (SP) organised in 14 Streaming-Multiprocessors (SM)

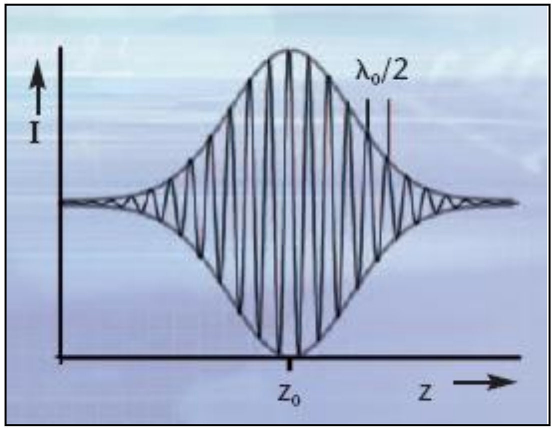


Examples: HPC tasks for optical 3D metrology on GPUs

- Application scenario: GPUs for optical metrology
  - Not Exa-scale computing
  - Using the cost-effective compute power of GPUs
  - Interest for a lot of SMEs (small and medium-sized enterprises)
- 3D measuring using white light interferometry



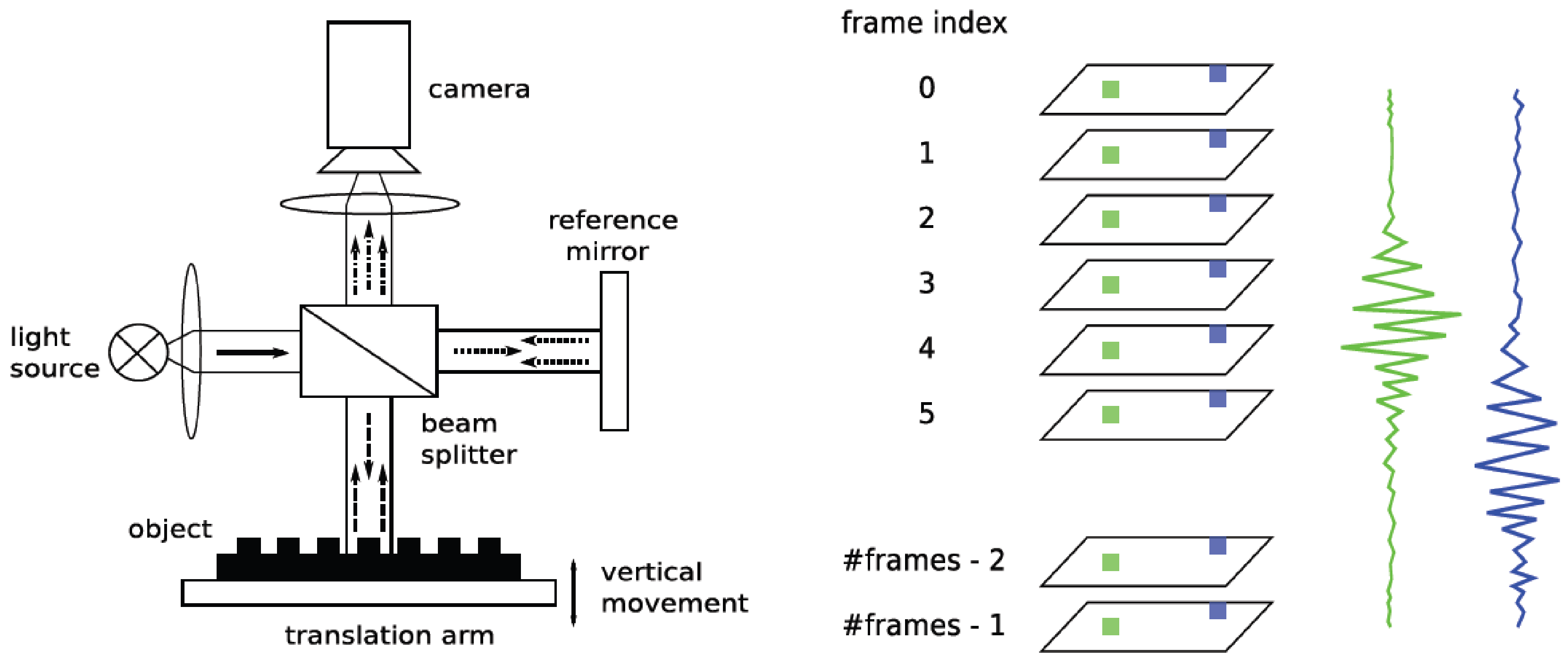
Twyman Green Interferometer



Interferogram

Examples: HPC tasks for optical 3D metrology on GPUs

- Huge amount of data
  - Up to 1000 layers with 1000 × 1000 pixels each



Examples: HPC tasks for optical 3D metrology on GPUs

- To filter out noise the so-called bucket-method has approved

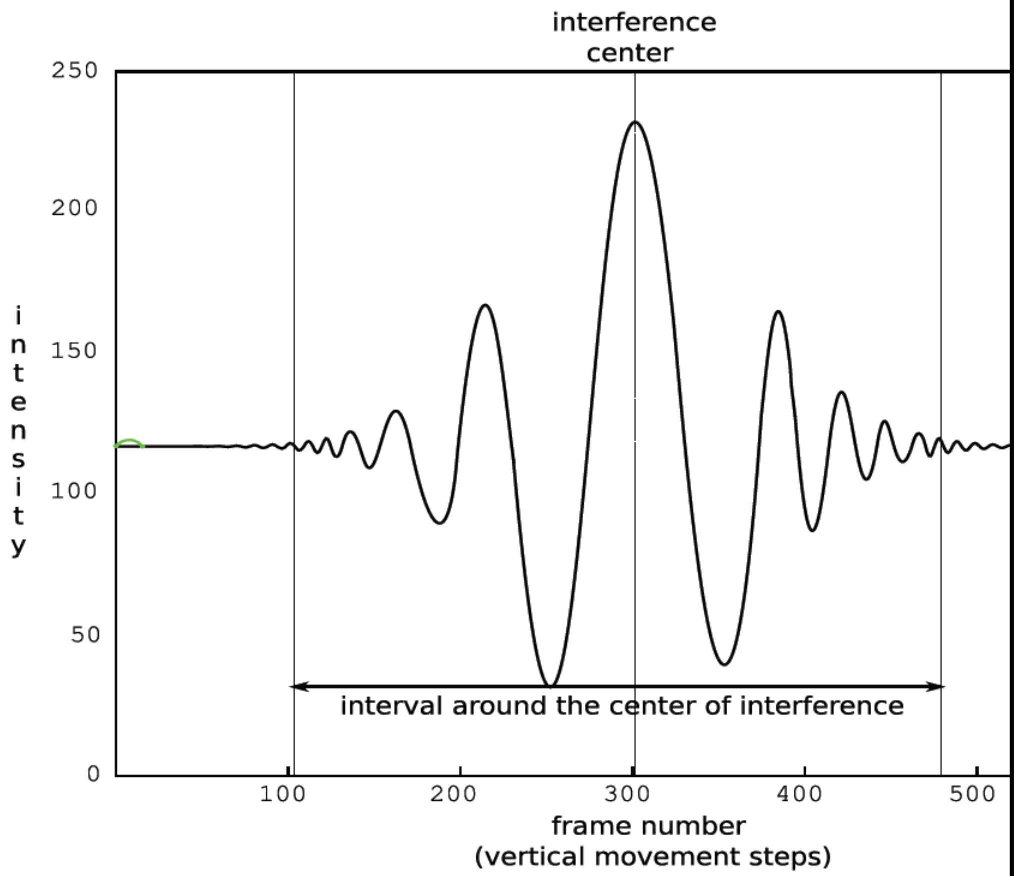
$$DF02(z) = I(z+0) - I(z+2) \tag{1}$$
$$DF13(z) = I(z+1) - I(z+3) \tag{2}$$
$$DF24(z) = I(z+2) - I(z+4) \tag{3}$$
$$B(z) = \frac{1}{2}\sqrt{DF13(z)^2 - DF02(z) \cdot DF24(z)} \tag{4}$$

- Exploiting symmetry reduces the amount of calculation

$$DF02(z_i) = DF13(z_{i-1}) \tag{5}$$
$$DF13(z_i) = DF24(z_{i-1}) \tag{6}$$
$$DF24(z_i) = I(z_i+2) - I(z_i+4) \tag{7}$$
$$B(z_i) = \frac{1}{2}\sqrt{DF24(z_{i-1})^2 - DF13(z_{i-1}) \cdot DF24(z_i)} \tag{8}$$

Examples: HPC tasks for optical 3D metrology on GPUs

- Illustration of the algorithm (i)

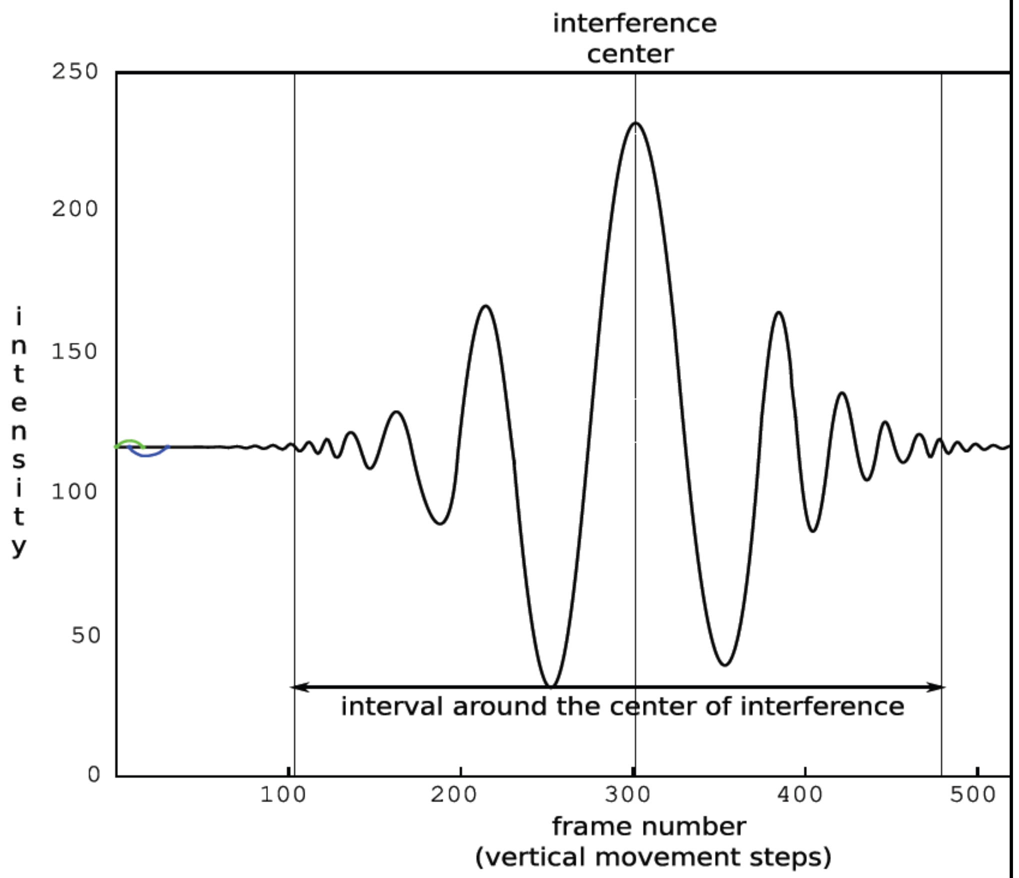


```
for(int i = 0; i < pixels; i++)
{
   diff02[i] = intens[0][i] - intens[2][i];
   diff13[i] = intens[1][i] - intens[3][i];
   max_bucket[i] = 0;
}

for(int l = 4; l < levels; l++)
{
   for(int i = 0; i < pixels; i++)
   {
     diff24[i] = intens[l - 2][i] - intens[l][i];
     cur_bucket = abs( diff13[i] * diff13[i] -
                       diff02[i] * diff24[i]) * 0.5;
     if(cur_bucket > max_bucket[i])
     {
       max_bucket[i] = cur_bucket;
       max_lvl[i] = l - 2;
     }
     diff02[i] = diff13[i];
     diff13[i] = diff24[i];
   }
}
```

Examples: HPC tasks for optical 3D metrology on GPUs

- Illustration of the algorithm (ii)

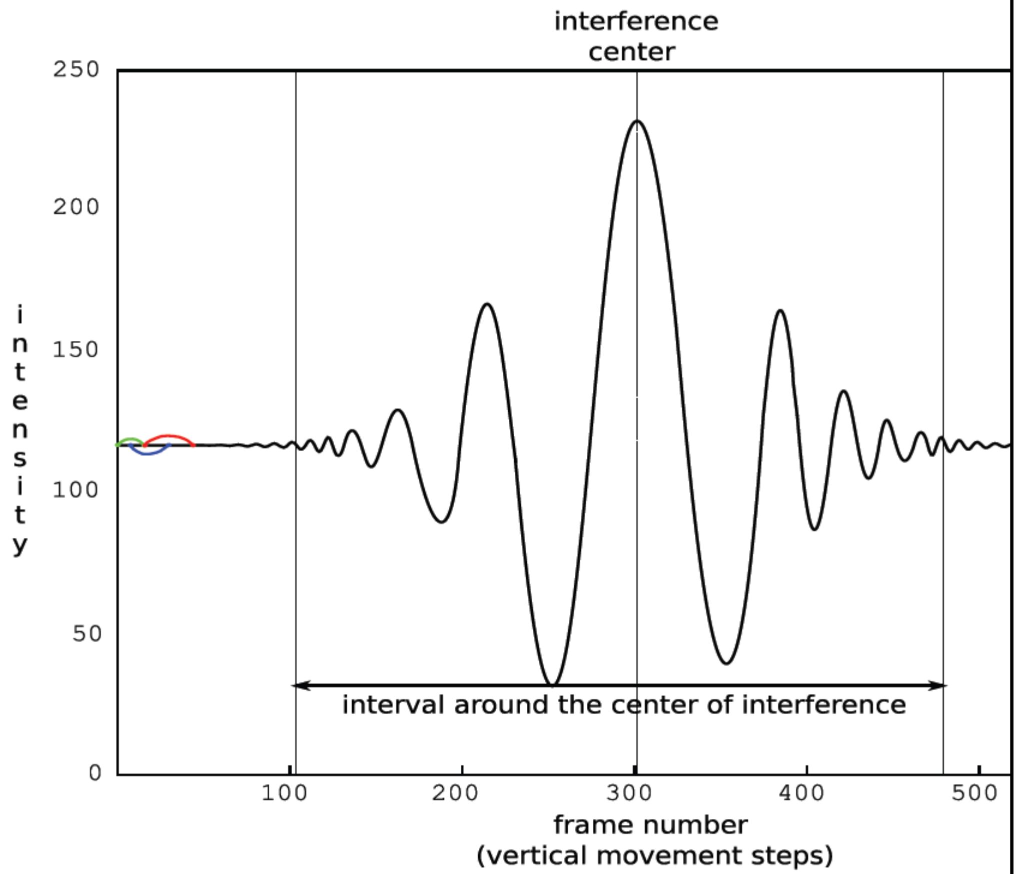


```
for(int i = 0; i < pixels; i++)
{
  diff02[i] = intens[0][i] - intens[2][i];
  diff13[i] = intens[1][i] - intens[3][i];
  max_bucket[i] = 0;
}

for(int l = 4; l < levels; l++)
{
  for(int i = 0; i < pixels; i++)
  {
    diff24[i] = intens[l - 2][i] - intens[l][i];
    cur_bucket = abs( diff13[i] * diff13[i] -
                      diff02[i] * diff24[i]) * 0.5;
    if(cur_bucket > max_bucket[i])
    {
      max_bucket[i] = cur_bucket;
      max_lvl[i] = l - 2;
    }
    diff02[i] = diff13[i];
    diff13[i] = diff24[i];
  }
}
```

Examples: HPC tasks for optical 3D metrology on GPUs

- Illustration of the algorithm (iii)

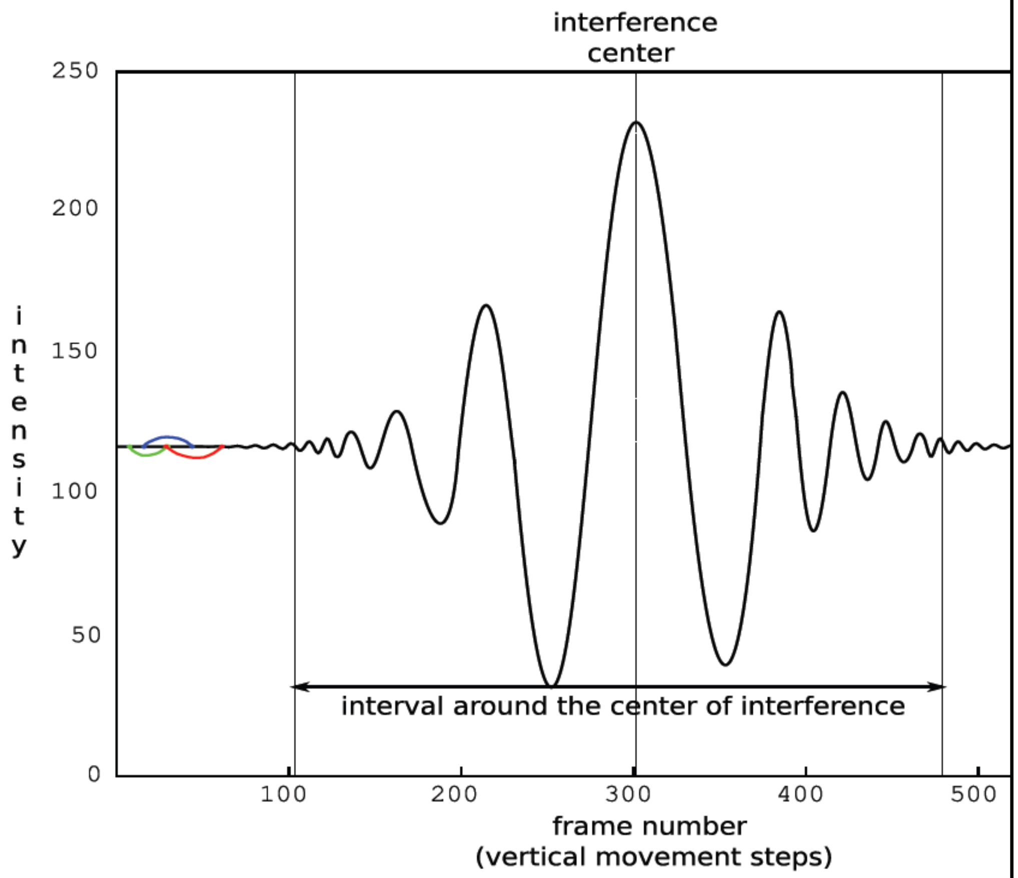


```
for(int i = 0; i < pixels; i++)
{
  diff02[i] = intens[0][i] - intens[2][i];
  diff13[i] = intens[1][i] - intens[3][i];
  max_bucket[i] = 0;
}

for(int l = 4; l < levels; l++)
{
  for(int i = 0; i < pixels; i++)
  {
    diff24[i] = intens[l - 2][i] - intens[l][i];
    cur_bucket = abs( diff13[i] * diff13[i] -
                      diff02[i] * diff24[i]) * 0.5;
    if(cur_bucket > max_bucket[i])
    {
      max_bucket[i] = cur_bucket;
      max_lvl[i] = l - 2;
    }
    diff02[i] = diff13[i];
    diff13[i] = diff24[i];
  }
}
```

Examples: HPC tasks for optical 3D metrology on GPUs

- Illustration of the algorithm (iv)

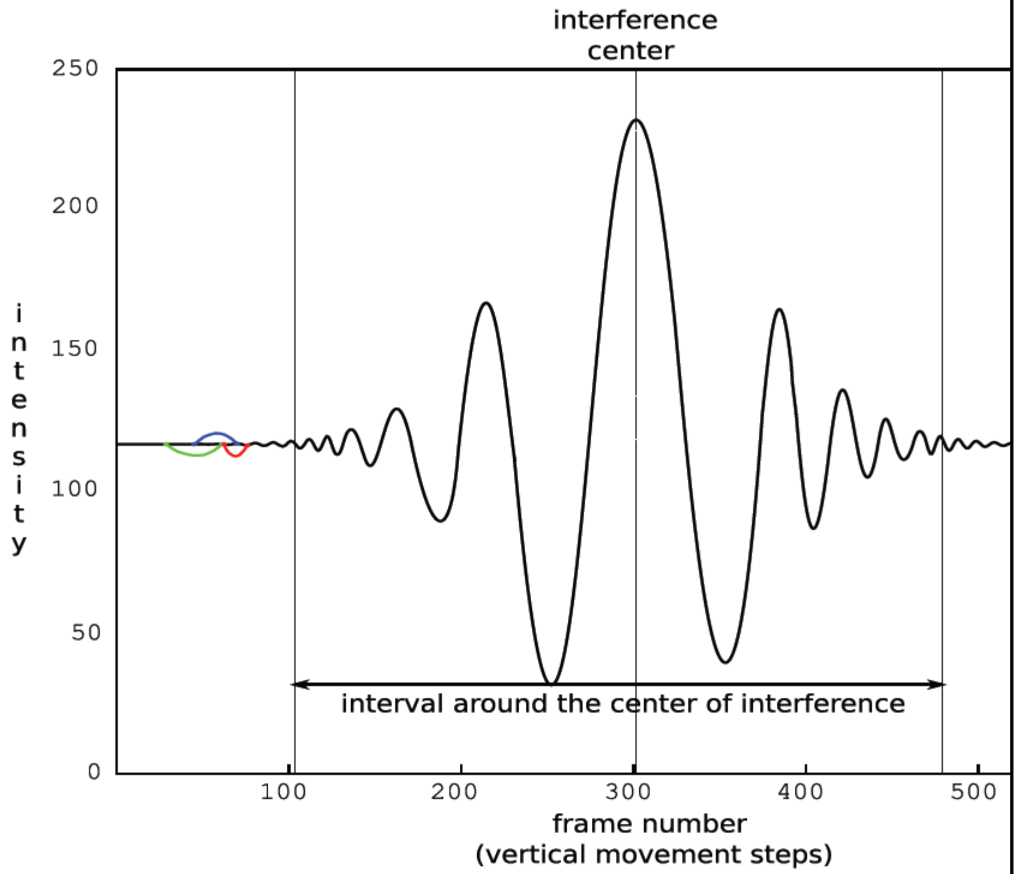


```
for(int i = 0; i < pixels; i++)
{
  diff02[i] = intens[0][i] - intens[2][i];
  diff13[i] = intens[1][i] - intens[3][i];
  max_bucket[i] = 0;
}

for(int l = 4; l < levels; l++)
{
  for(int i = 0; i < pixels; i++)
  {
    diff24[i] = intens[l - 2][i] - intens[l][i];
    cur_bucket = abs( diff13[i] * diff13[i] -
                      diff02[i] * diff24[i]) * 0.5;
    if(cur_bucket > max_bucket[i])
    {
      max_bucket[i] = cur_bucket;
      max_lvl[i] = l - 2;
    }
    diff02[i] = diff13[i];
    diff13[i] = diff24[i];
  }
}
```

Examples: HPC tasks for optical 3D metrology on GPUs

- Illustration of the algorithm (v)

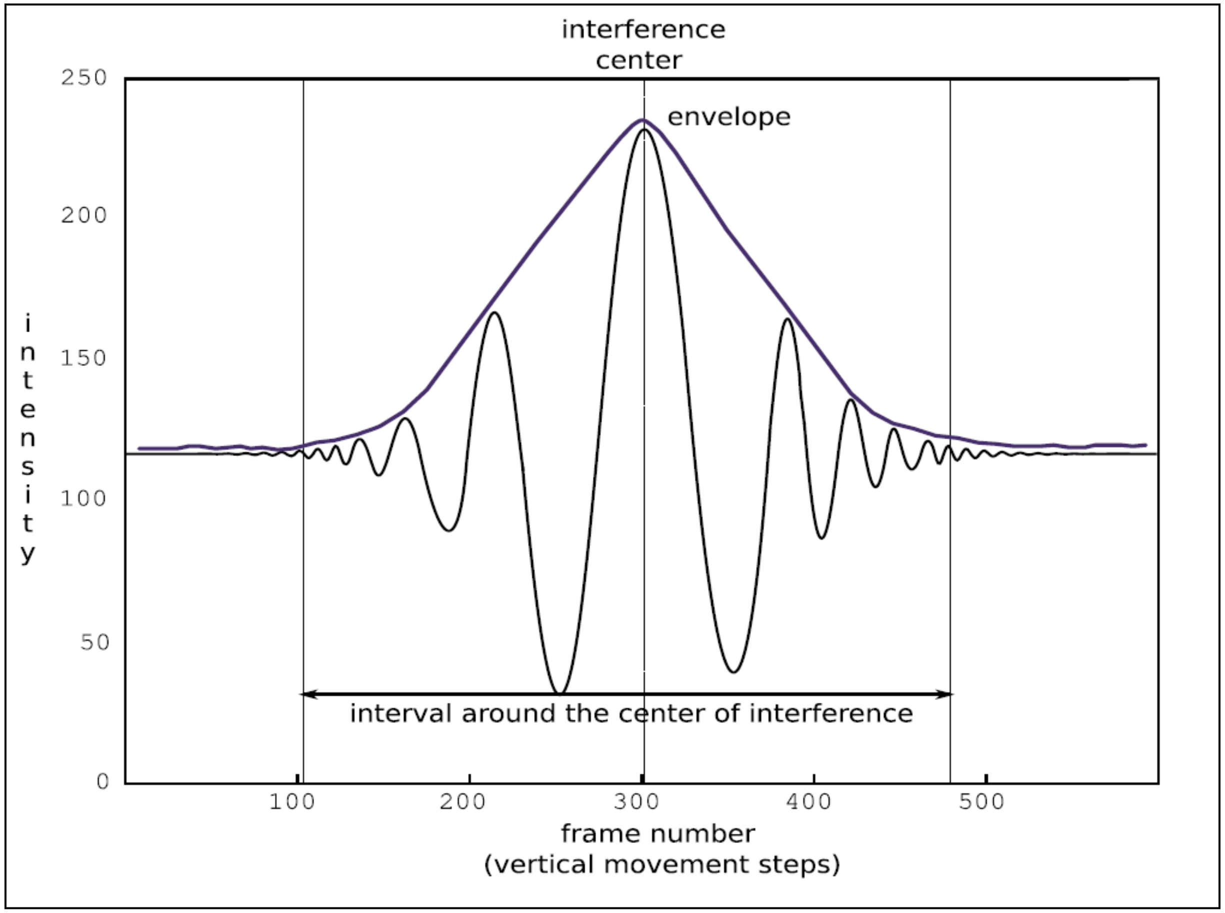


```
for(int i = 0; i < pixels; i++)
{
  diff02[i] = intens[0][i] - intens[2][i];
  diff13[i] = intens[1][i] - intens[3][i];
  max_bucket[i] = 0;
}

for(int l = 4; l < levels; l++)
{
  for(int i = 0; i < pixels; i++)
  {
    diff24[i] = intens[l - 2][i] - intens[l][i];
    cur_bucket = abs( diff13[i] * diff13[i] -
                      diff02[i] * diff24[i]) * 0.5;
    if(cur_bucket > max_bucket[i])
    {
      max_bucket[i] = cur_bucket;
      max_lvl[i] = l - 2;
    }
    diff02[i] = diff13[i];
    diff13[i] = diff24[i];
  }
}
```

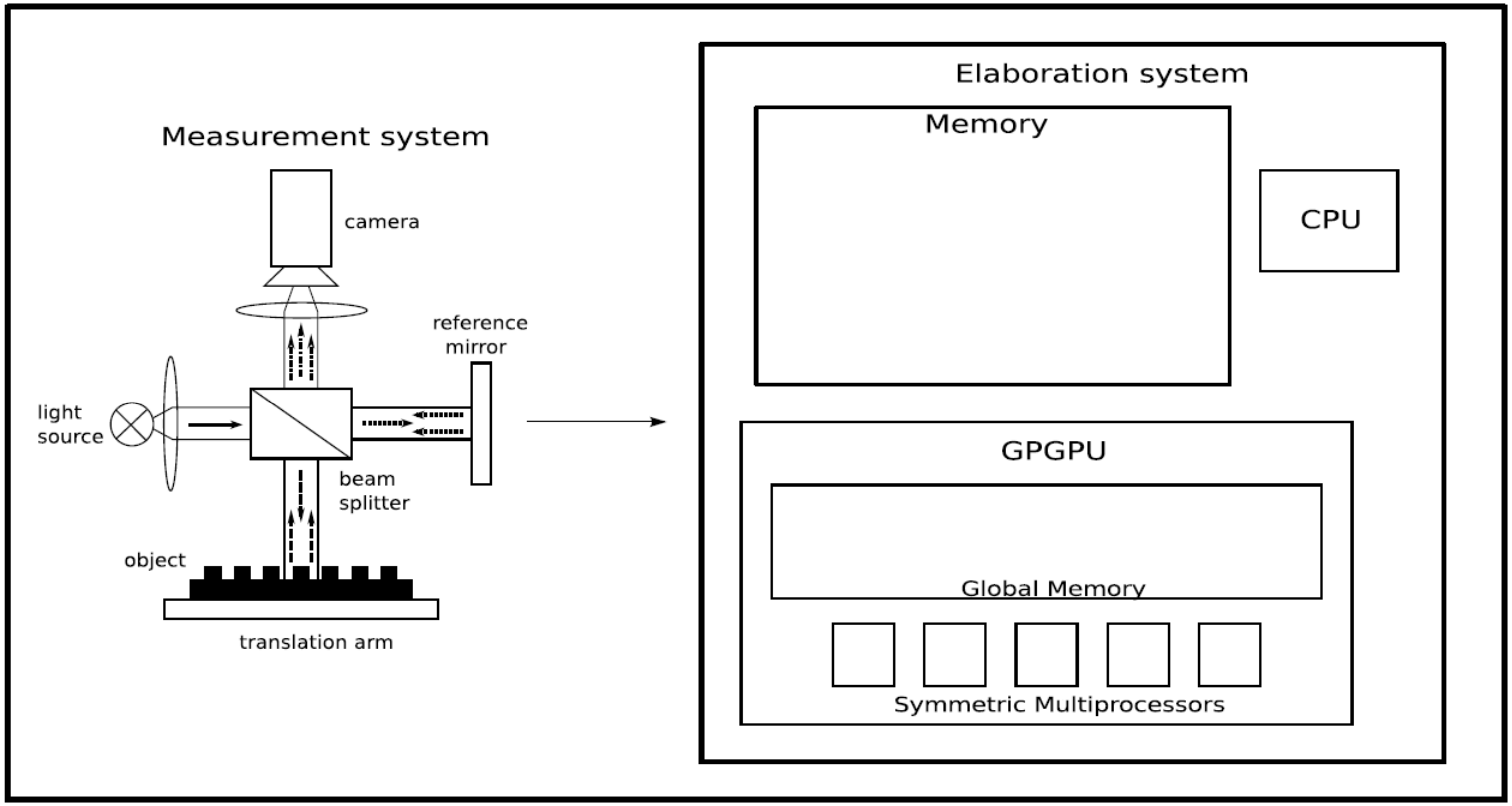
Examples: HPC tasks for optical 3D metrology on GPUs

- Finally the envelope has to be found



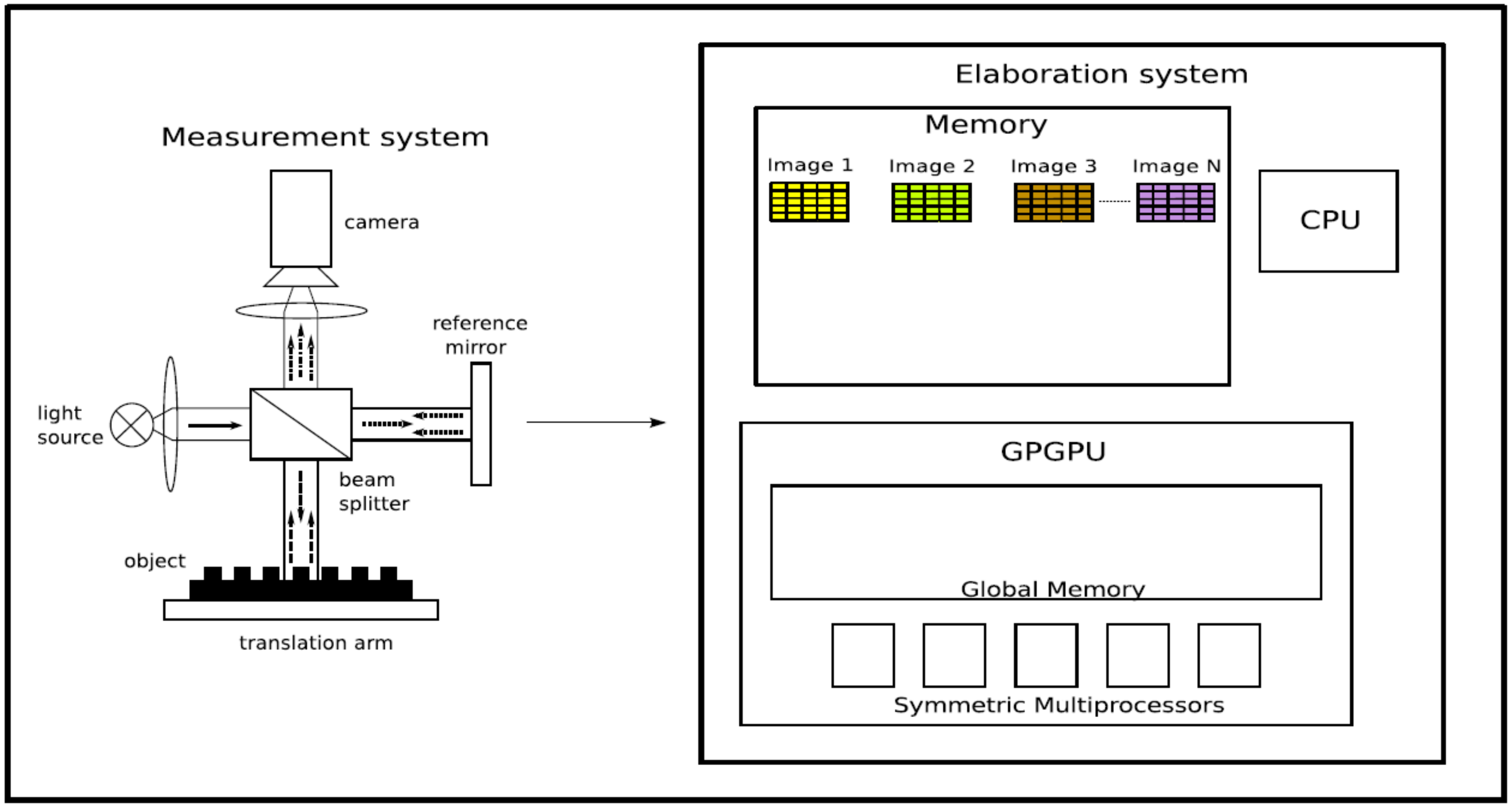
Examples: HPC tasks for optical 3D metrology on GPUs

- How to bring that solution in an efficient way onto a GPGPU?



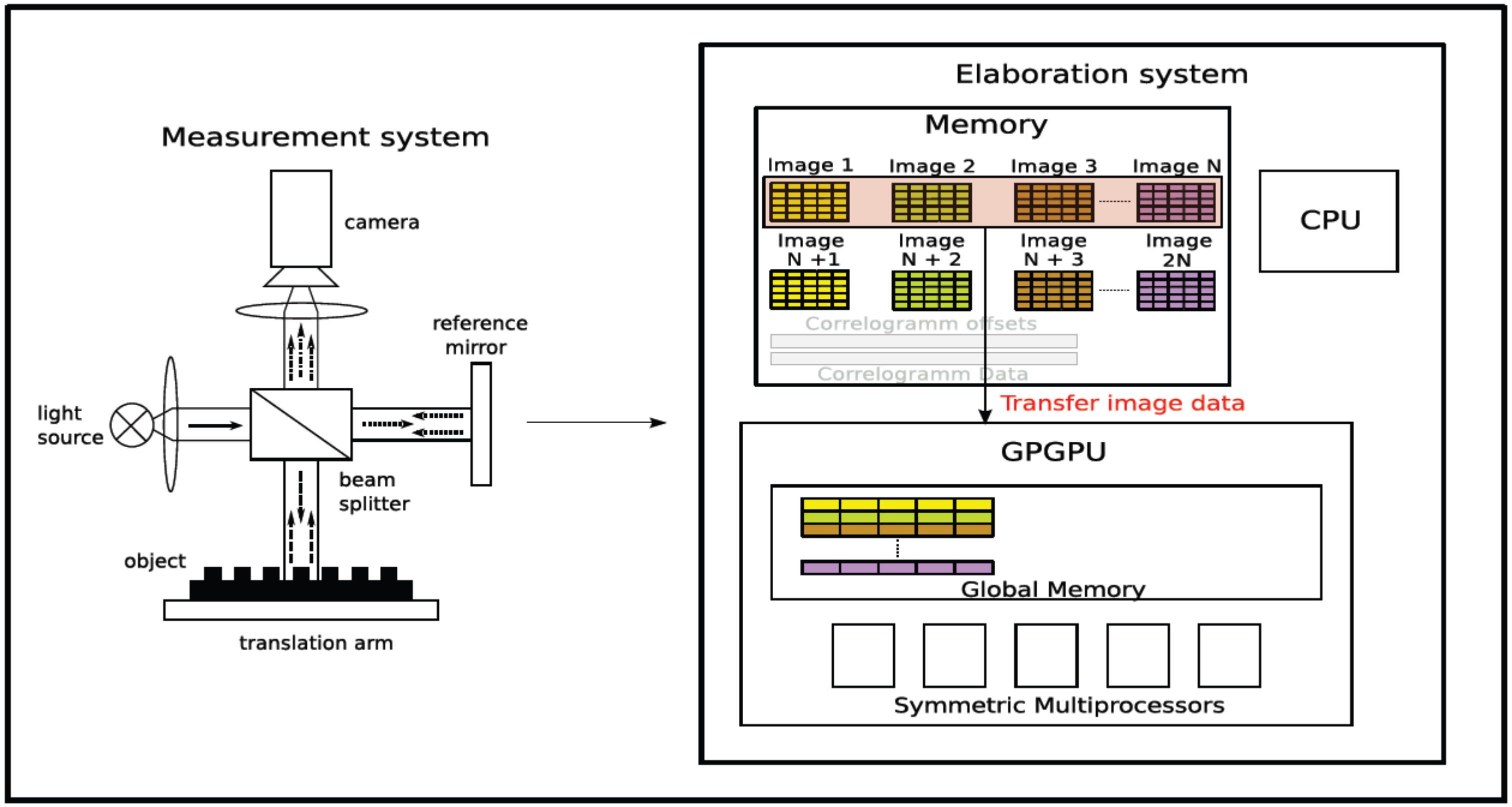
Examples: HPC tasks for optical 3D metrology on GPUs

- Reading in the input data from the measurement system into the host‘s main memory



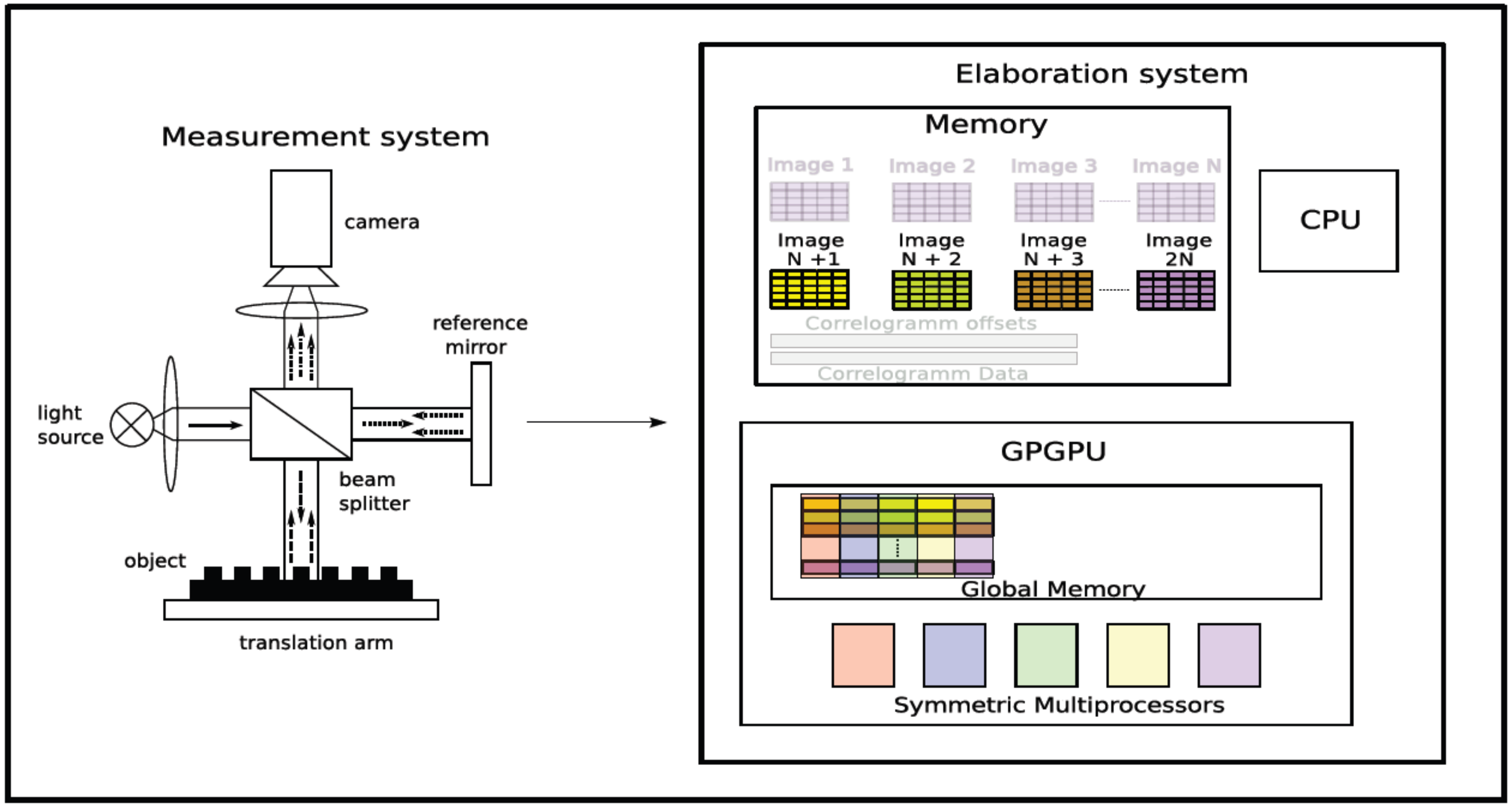
Examples: HPC tasks for optical 3D metrology on GPUs

- Transferring the image data from host memory to GPU memory



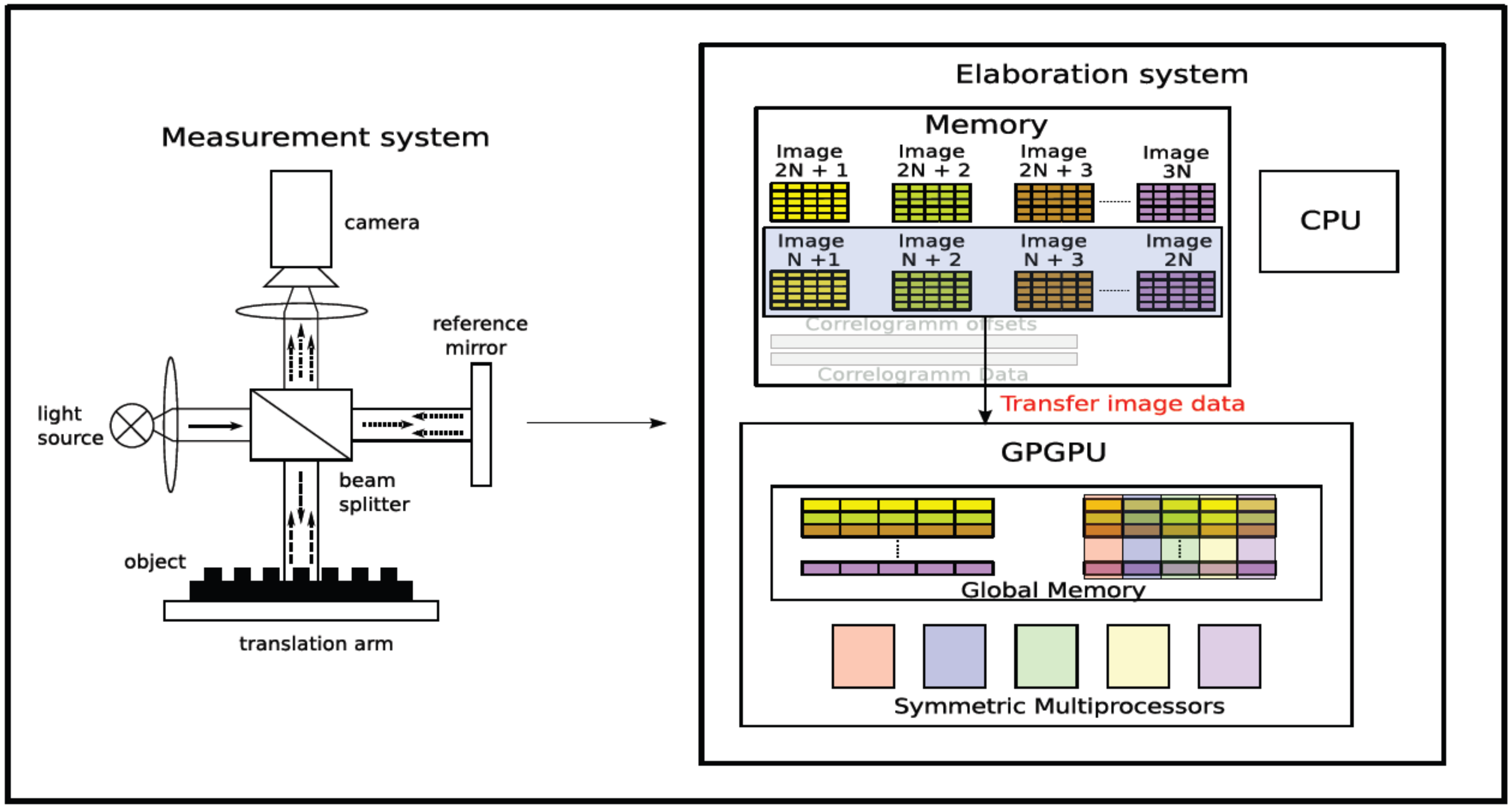
Examples: HPC tasks for optical 3D metrology on GPUs

- The symmetric multiprocessors start to work by accessing global and local memory



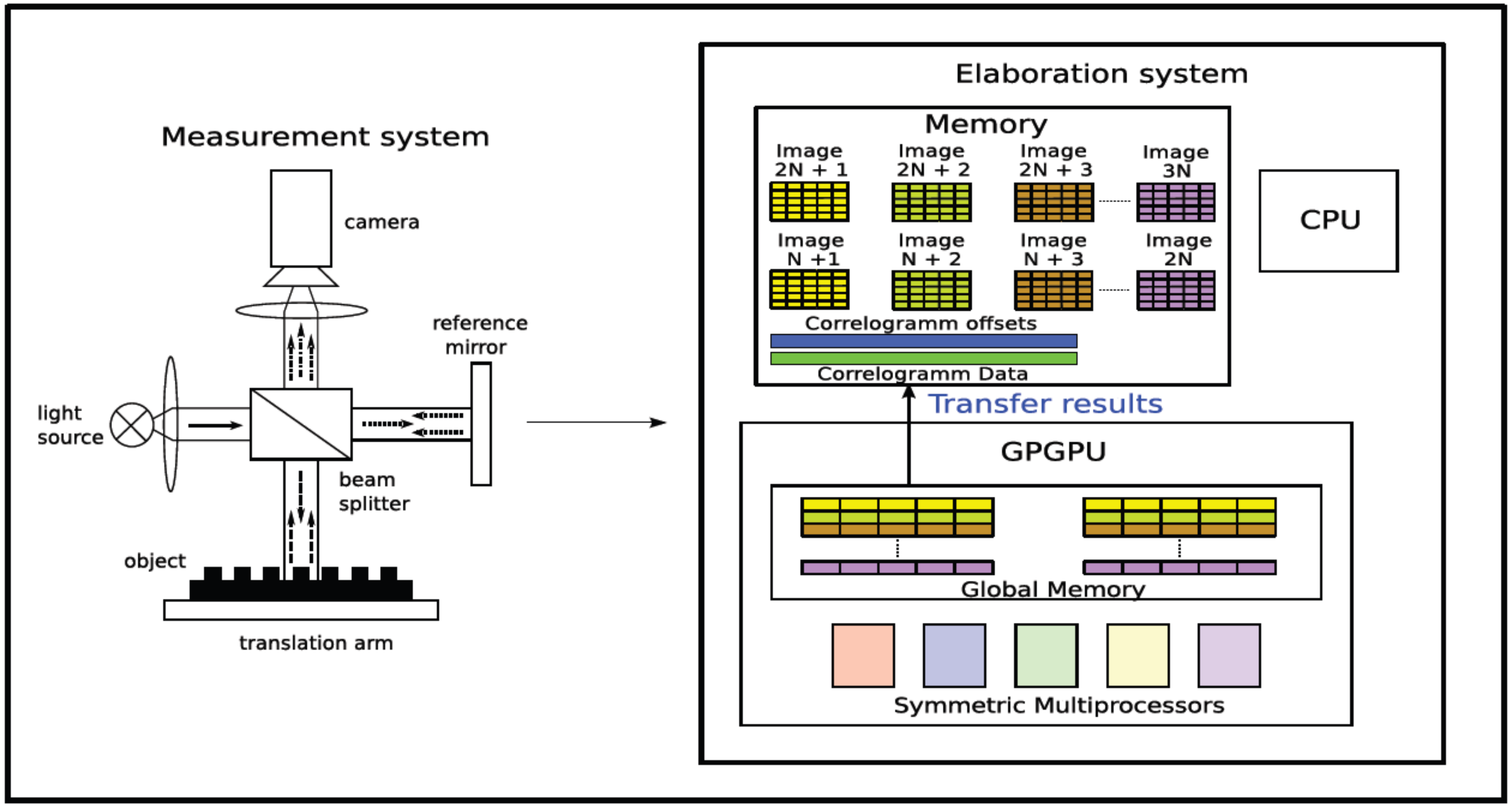
Examples: HPC tasks for optical 3D metrology on GPUs

- Concurrently to calculation the next layers are transferred to GPU memory



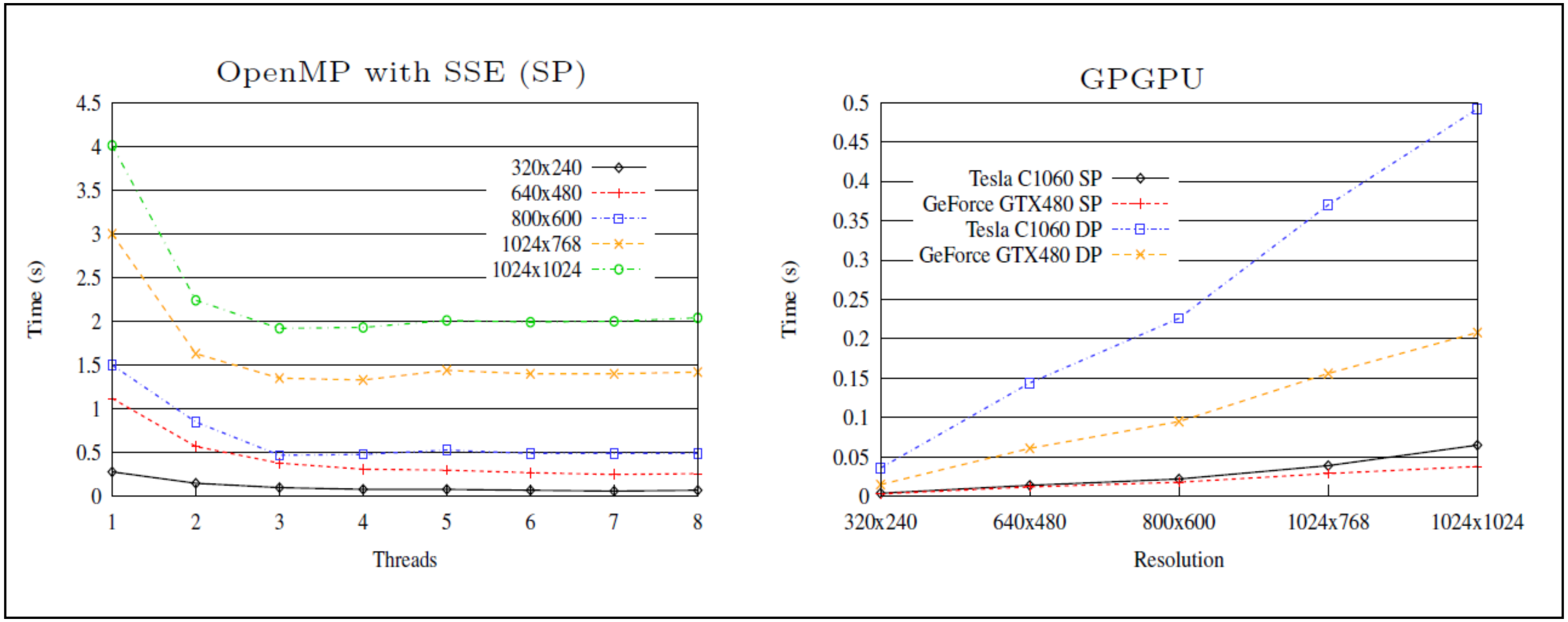
Examples: HPC tasks for optical 3D metrology on GPUs

- Finally the results are transferred back to main memory



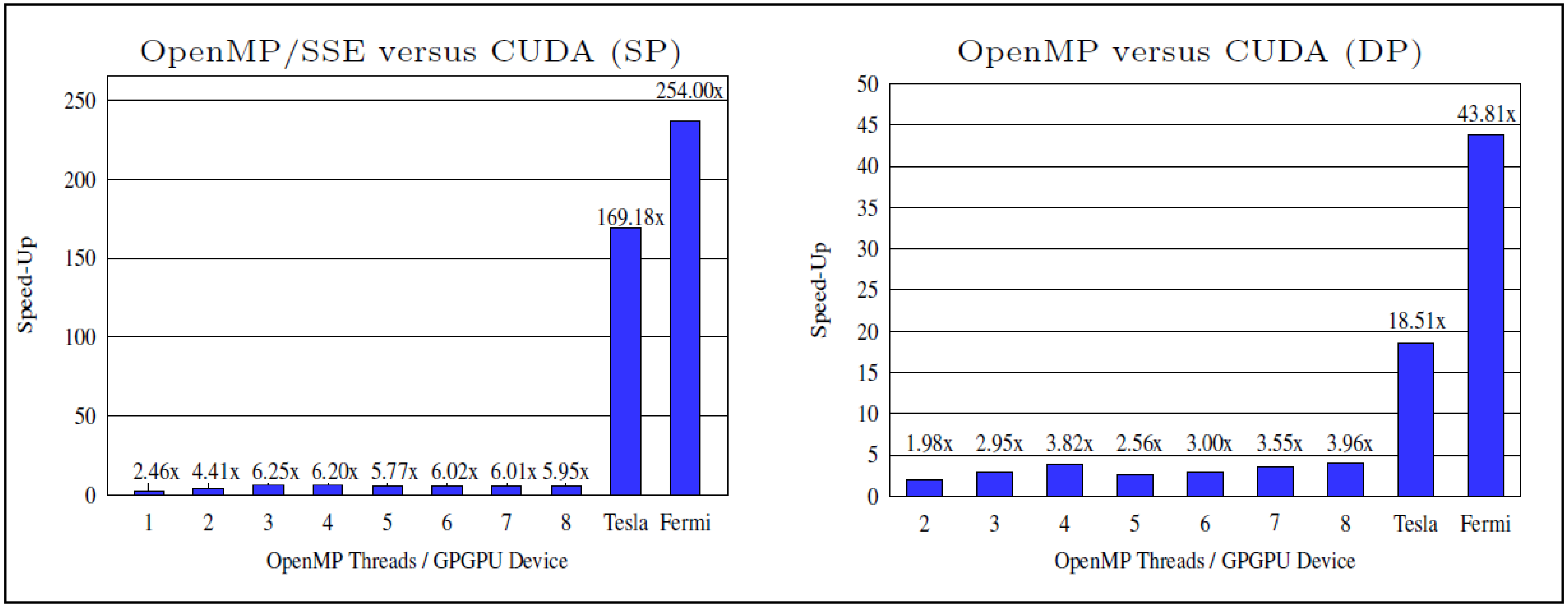
Examples: HPC tasks for optical 3D metrology on GPUs

- Perfomance comparison with GPU and normal CPU



Examples: HPC tasks for optical 3D metrology on GPUs

- Expressed using Speed-up values



Performance modelling – Roofline model

A simple two-dimensional performance model

- Main reason for bottleneck
  - die „off-chip“ memory bandwidth
  - Request for a model
    - Relation between processor performance and off-chip memory traffic

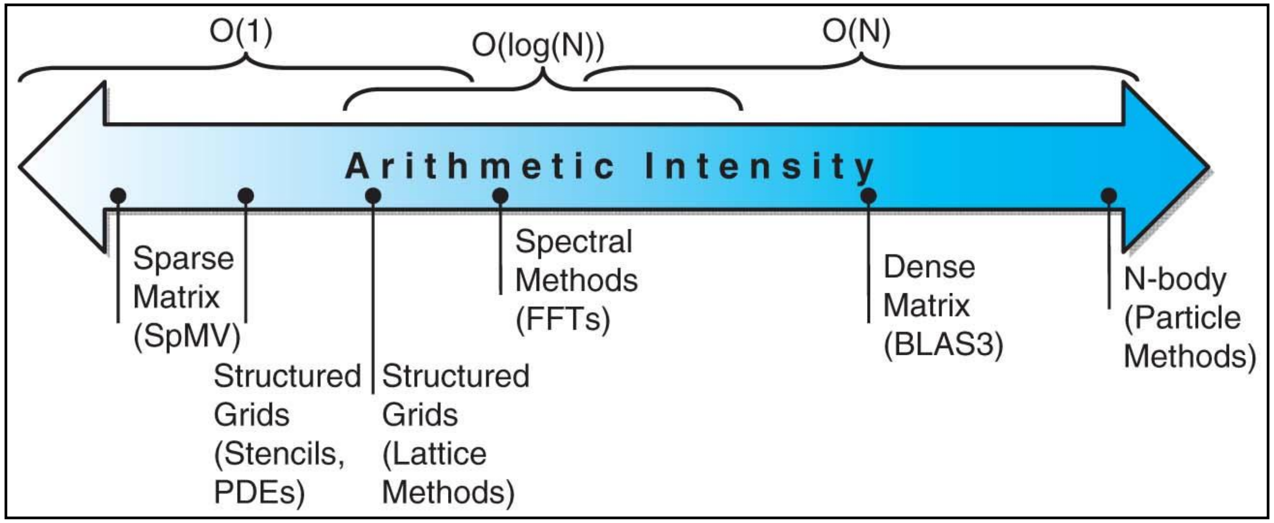
- Important value: operational intensity
  - Number of operations per fetched byte [Flops / Byte]
  - Measurement parameter for traffic between DRAM memory and caches
  - Not between caches and processor → Arithmetic Intensity

- Roofline model
  - 2D model, which unifies operational intensity, memory bandwidth and maximum achievable computing performance

Performance modelling – Roofline model

Determine the operational intensity for kernels

- Kernel:
  - SparseMatrix, Structured Grids (Stencils), Structured Grids (Lattice methods), spectral methods (Fast Fourier Transformations – FFT, Dense matrix, N-body problems)
  - Scaling depends on O(N), O(log(N), resp. It is independent from the problem size O(1)



Performance modelling – Roofline model

Arithmetical / operational intensity

- Both can be measured in [Flops / Byte]
- Derivable the necessary bandwidth for the memory system
  - Quotient of achievable peak floating point performance / Operational resp. arithmetic intensity

$$\frac{Floating-PointOperations/Sec}{Floating-PointOperations/Byte} = \frac{Bytes}{Sec}$$

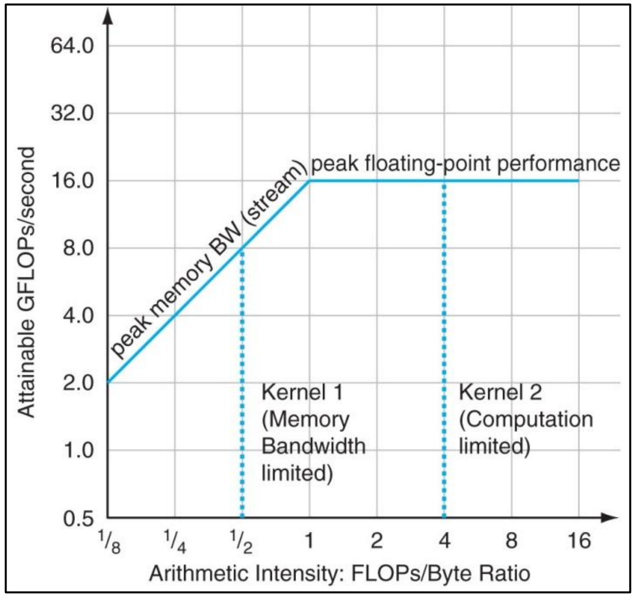
- The real achievable computing performance

$$Attainable\ GLOP/sec = Min(Peak\ Floating-Point\ Performance,\ Peak\ Memory\ Bandwidth \times Operational\ Intensity)$$

Performance modelling – Roofline model

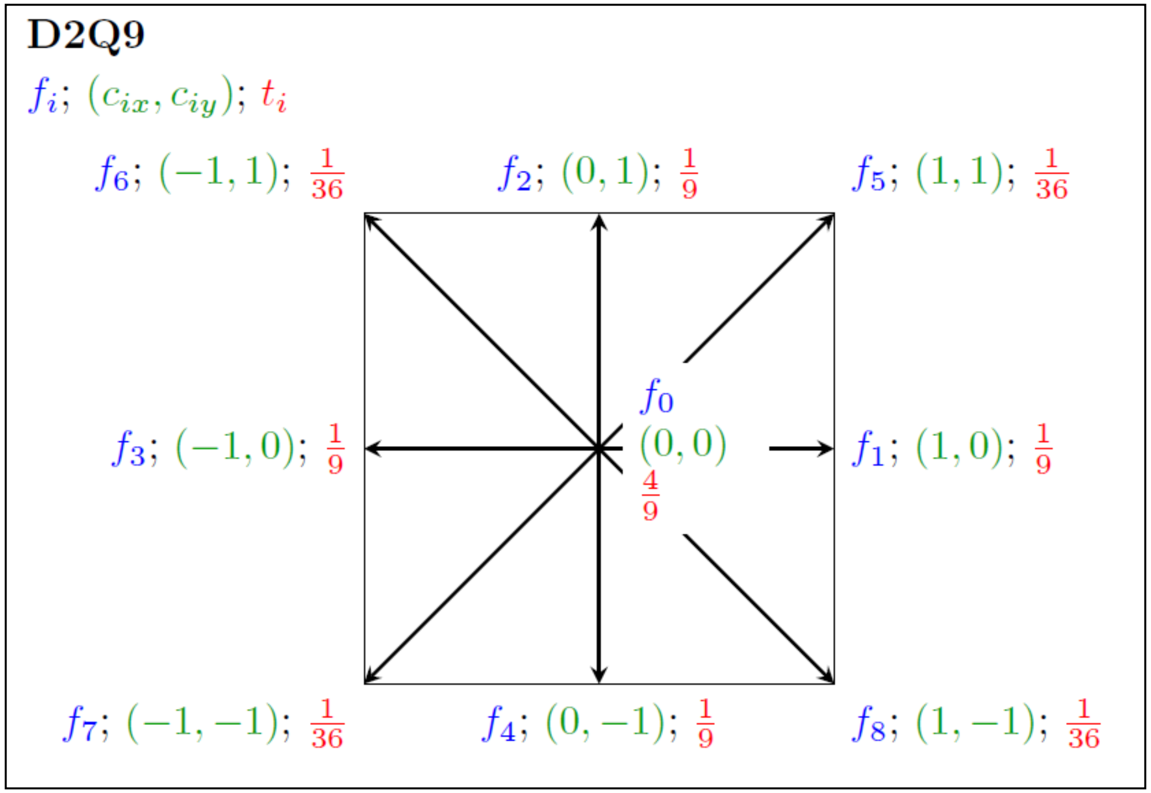
Example: Graphical presentation of the roofline model

- Opteron X2 , Dual Core @ 2 GHz
  - Kernel 1 is memory bandwidth bound
    - 0.5 FLOPSs / Byte; memory bandwidth limited; limits computing performance to 8 GFLOPs / sec
  - Kernel 2 is computation bounded or limited
    - 4 FLOPs / Byte; memory bandwidth is not the problem (max. 16 GB/s)



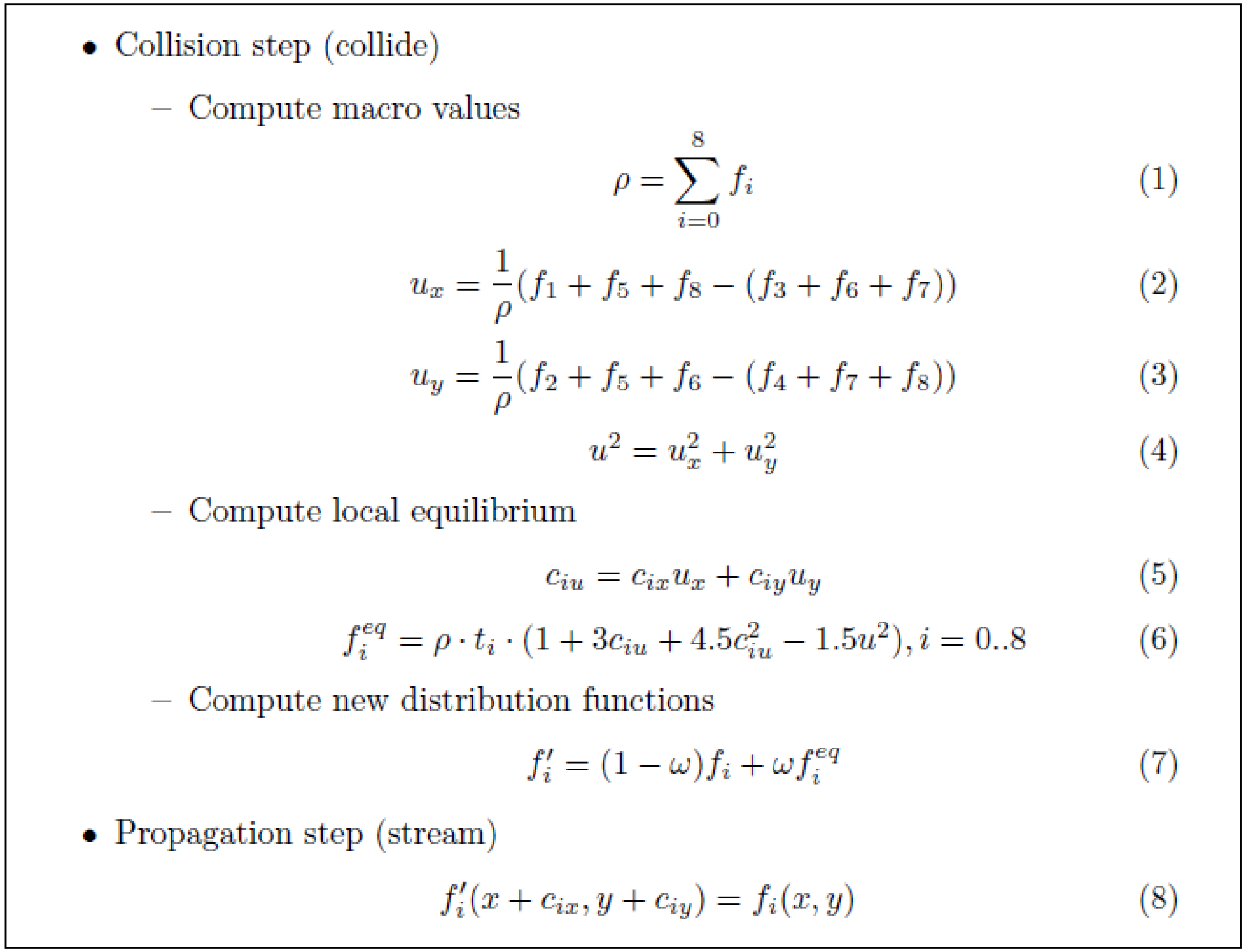
Examples: HPC kernels on FPGAs

e.g. Lattice-Boltzman method (LBM)


D2Q9
f_i; (c_{ix}, c_{iy}); t_i
f_6; $(-1, 1)$; $\frac{1}{36}$
f_2; $(0, 1)$; $\frac{1}{9}$
f_5; $(1, 1)$; $\frac{1}{36}$
f_3; $(-1, 0)$; $\frac{1}{9}$
f_0 $(0, 0)$ $\frac{4}{9}$
f_1; $(1, 0)$; $\frac{1}{9}$
f_7; $(-1, -1)$; $\frac{1}{36}$
f_4; $(0, -1)$; $\frac{1}{9}$
f_8; $(1, -1)$; $\frac{1}{36}$

Examples: HPC kernels on FPGAs

The compute kernel of the LBM



- Collision step (collide)
  - Compute macro values

$$\rho = \sum_{i=0}^{8} f_i \tag{1}$$

$$u_x = \frac{1}{\rho}(f_1 + f_5 + f_8 - (f_3 + f_6 + f_7)) \tag{2}$$

$$u_y = \frac{1}{\rho}(f_2 + f_5 + f_6 - (f_4 + f_7 + f_8)) \tag{3}$$

$$u^2 = u_x^2 + u_y^2 \tag{4}$$

  - Compute local equilibrium

$$c_{iu} = c_{ix}u_x + c_{iy}u_y \tag{5}$$

$$f_i^{eq} = \rho \cdot t_i \cdot (1 + 3c_{iu} + 4.5c_{iu}^2 - 1.5u^2), i = 0..8 \tag{6}$$

  - Compute new distribution functions

$$f_i' = (1-\omega)f_i + \omega f_i^{eq} \tag{7}$$

- Propagation step (stream)

$$f_i'(x + c_{ix}, y + c_{iy}) = f_i(x, y) \tag{8}$$

Examples: HPC kernels on FPGAs

- Computational Complexity
  - after optimization: 95 floating-point operations
    - 52 additions(+)
    - 42 multiplications (x)
    - 1 inversion (1/x)
- 1 Msu/s = 95 MFLOPS

Examples: HPC kernels on FPGAs

- Memory Requirements
  - Single precision floating-point (32 bit): 36 Bytes/site
  - Symmetrical read/write requirements
  - Pull- or push- implementation of the propagation step:
    - pull: stream and collide: read from neighbor cells, write to local
    - push: collide and stream: read from local cell, write to neighbors
- 1 Msu/s = 36 MB/s (read and write, 72 MB/s combined)

Examples: HPC kernels on FPGAs

Required ressources

Megafunction	Combinational ALUTs	Dedicated Logic Registers	DSP blocks(9×9)	Latency	f_{max} (MHz)
ADD_SUB	141	338..850	—	7..14	218..416
MUL	55	136..389	8	5,6,10,11	240..466
INV	391	765	32	20	427

Examples: HPC kernels on FPGAs

Assuming the data is stored in host memory

achievable performance ranges assuming memory-bounding

- from 5.6 Msu/s (for PCIe x1, Gen.1)
- upto 80 Msu/s (for PCIe x8, Gen.2)

Comparison data

K. Sano, Custom Computing with Reconfigurable Technologies for Efficient Acceleration of CFD Kernels, ParCFD 2010

- Pentium4 at 3.4GHz: 7.0..8.9 Msu/s
- Opteron at 2.2GHz: 10 Msu/s
- FPGA (Xilinx Virtex-4 at 100 MHz) 26 Msu/s for grids of size 1.0×10^5 grid points

Another example: 2D Laplace equation on FPGAs

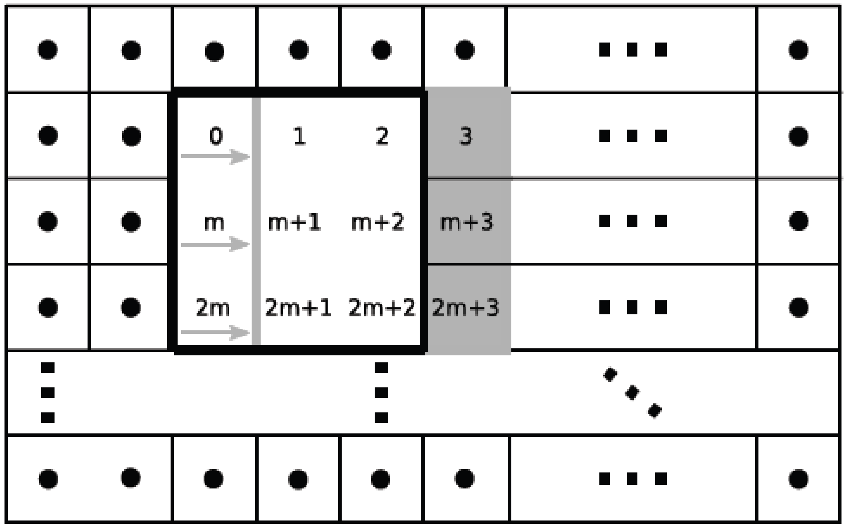
The compute kernel of 2D heat equation

$$\frac{\partial T}{dt} = k \cdot \left(\frac{\partial^2 T}{\partial^2 x} + \frac{\partial^2 T}{\partial^2 y} \right)$$

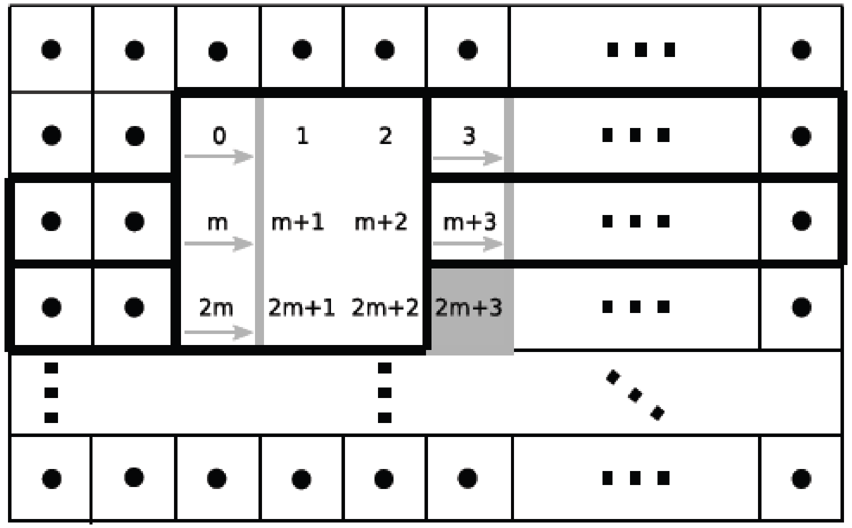
$$\Rightarrow T_{i,j,t+1} = T_{i,j,t} +$$
$$0.25 \times (T_{i-1,j,t} + T_{i+1,j,t} + T_{i,j+1,t} + T_{i,j-1,t} - 4 \times T_{i,j,t})$$

Examples: HPC kernels on FPGAs

- Partial buffering vs. full buffering
  - Sliding window operation (SWO)



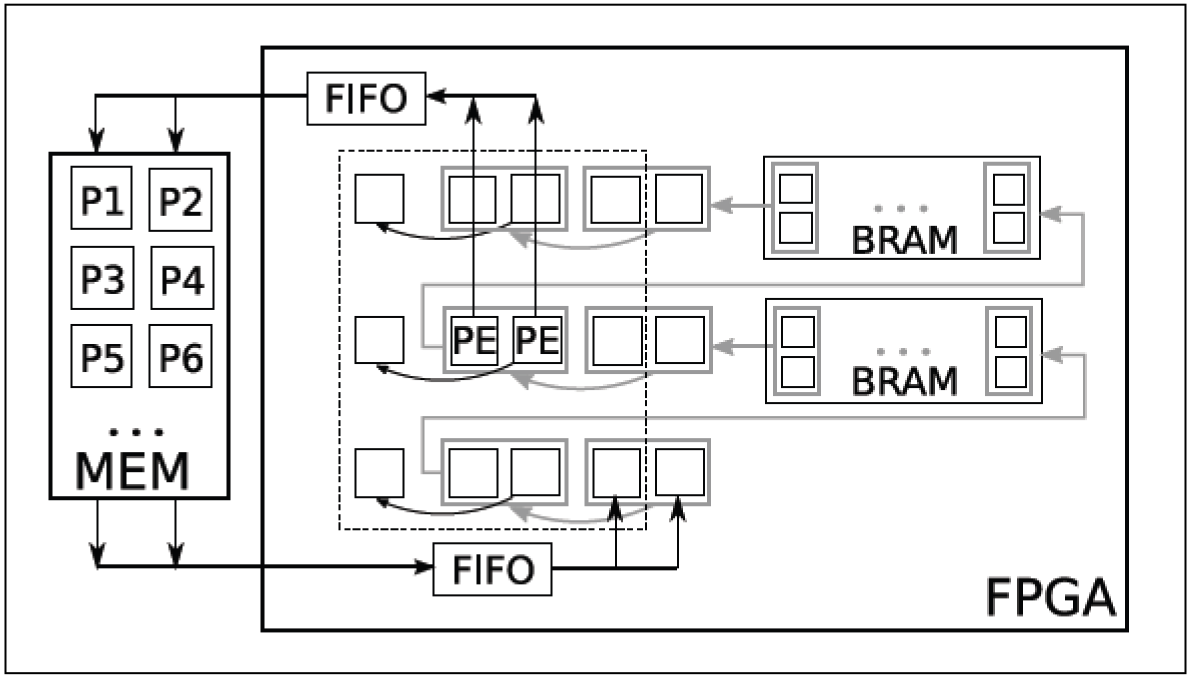
(a) Partial Buffering



(b) Full Buffering

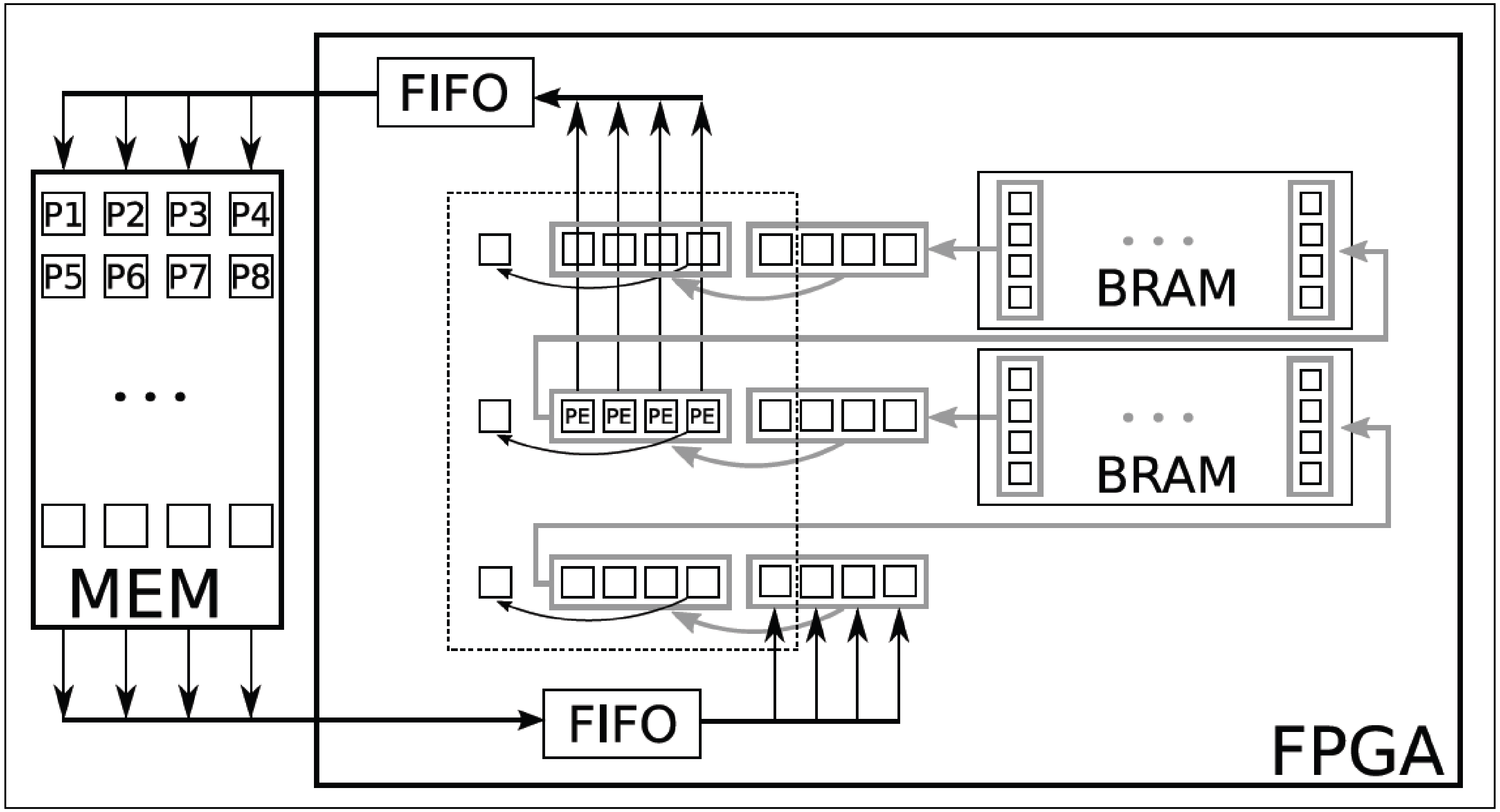
Examples: HPC kernels on FPGAs

Expand full buffering to 2 processors in FPGA



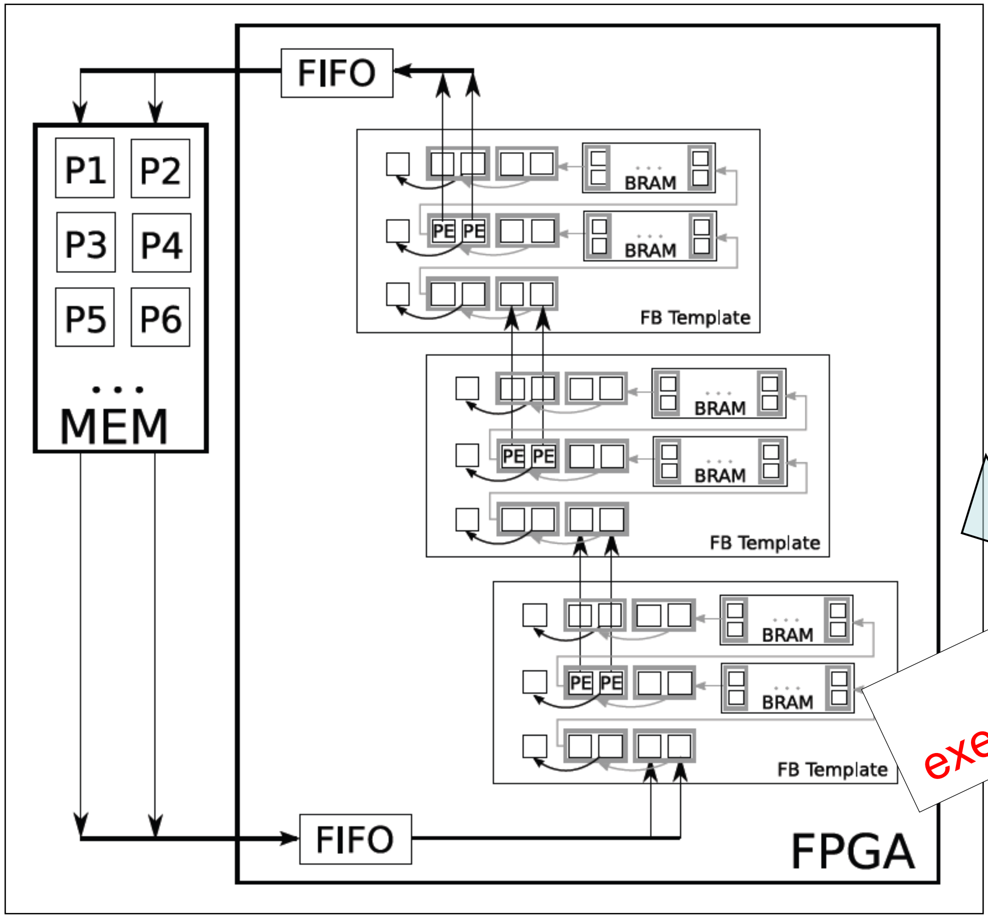
Examples: HPC kernels on FPGAs

Expand full buffering to 4 processors in FPGA



Examples: HPC kernels on FPGAs

Combining SWO with loop unrolling leads to further improvement

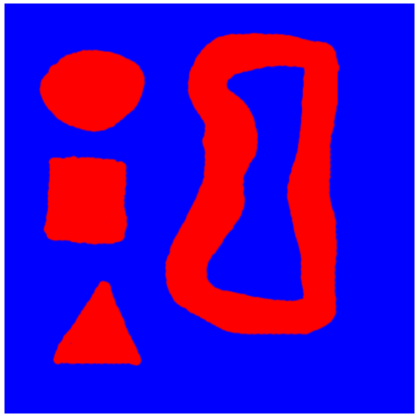


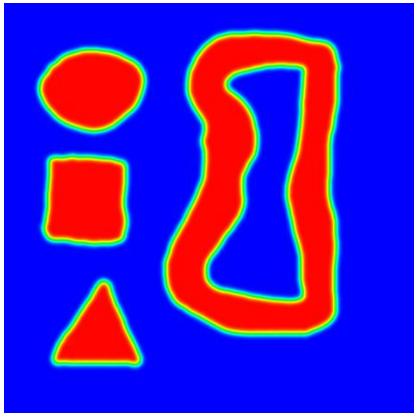
```
....
for $(i=1; i<N; i++)$ {
    for $(j=1; j<M; j++)$ {
        $intermediate = K \times (T_{i-1,j,t} + T_{i+1,j,t} + T_{i,j+1,t} + T_{i,j-1,t} - 4 \times T_{i,j,t});$
        $T_{i,j,t+1} = T_{i,j,t} + intermediate;$
    }
}
....
```

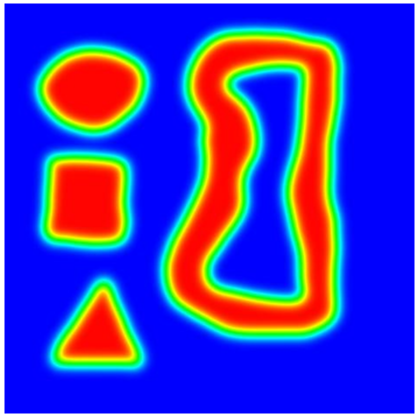
Linearize loop and execute parts in pipeline mode

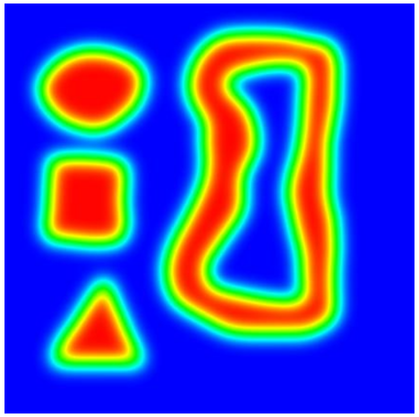
Examples: HPC kernels on FPGAs

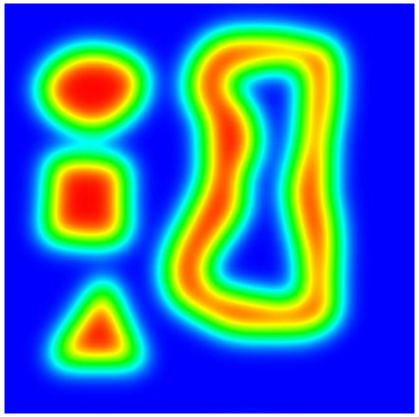
Proof by simulation that the approach is effective

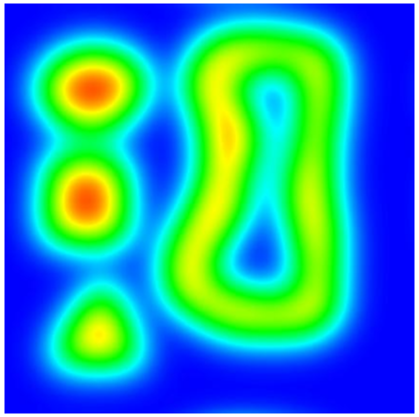

i = 0

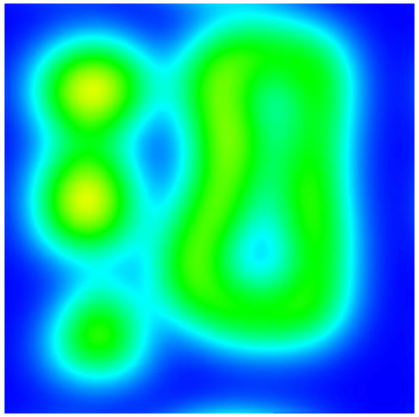

i = 100


i = 500


i = 1000


i = 2000


i = 5000


i = 10000

Examples: HPC kernels on FPGAs

Synthesis results

- Linear scaling
- Performance limited by number of ressources

$$\boxed{\begin{aligned} S_{it} &= \frac{C_{serial}}{C_{total}} = \frac{m \cdot n \cdot I_{total}}{C_{total}} \\ &= \frac{m \cdot n \cdot I_{total} \cdot p \cdot it}{m \cdot n \cdot I_{total} + it^2(m + 2 \cdot p)} \approx p \cdot it \end{aligned}}$$

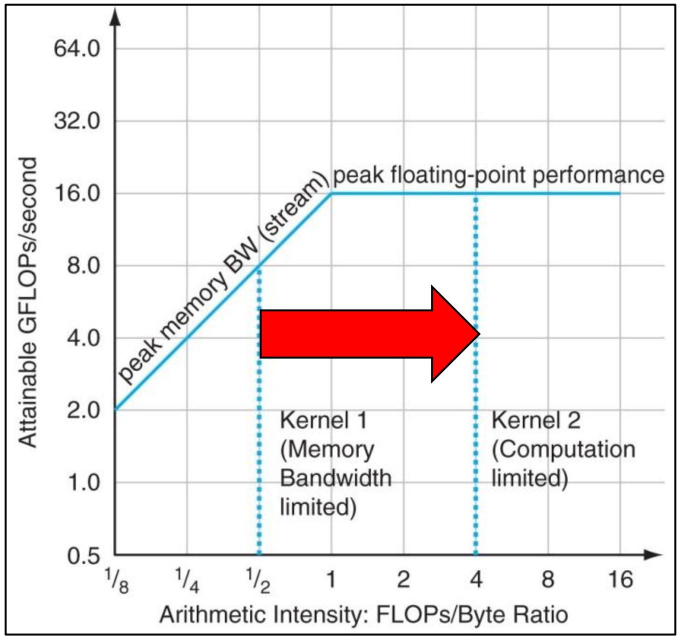
SYNTHESIS RESULTS FOR THE HEAT TRANSFER SIMULATION

p	it	LUTs	Registers	BRAMs	Speedup	Time (s)
1	1	0.4k	0.3k	2	1	52.4
2	1	0.7k	0.5k	4	2	26.2
2	8	5.5k	3.9k	32	16	3.2
2	16	11.0k	7.8k	64	32	1.6
2	32	22.1k	15.6k	128	64	0.8

Examples: HPC kernels on FPGAs

Shift the operational intensity to the right in the roofline model

- By applying loop unrooling and full buffering
- More operations can be carried out on the same amount of fetched data from memory
- Peak performance of your system can be reached



Examples: HPC kernels on FPGAs

Lecture: Architecture of Supercomputers

- Basic principles
  - Roofline model
  - Performance analysis of parallel computers
  - Processing in modern homogenous and heterogeneous multi-core architectures
  - GPU architecture and programming
  - Pipeline processing
  - Symmetrical Multi-Threading
  - High-Performance Networks

- Applied in real architectures
  - Earth Simulator
  - IBM Blue Gene
  - IBM Roadrunner