Dresden Microelectronics Academy

Orchestration: Turning material breakthroughs into application performance

Jeronimo Castrillon

(on behalf of the Orchestration team)

TU Dresden – Cfaed

Chair for Compiler Construction

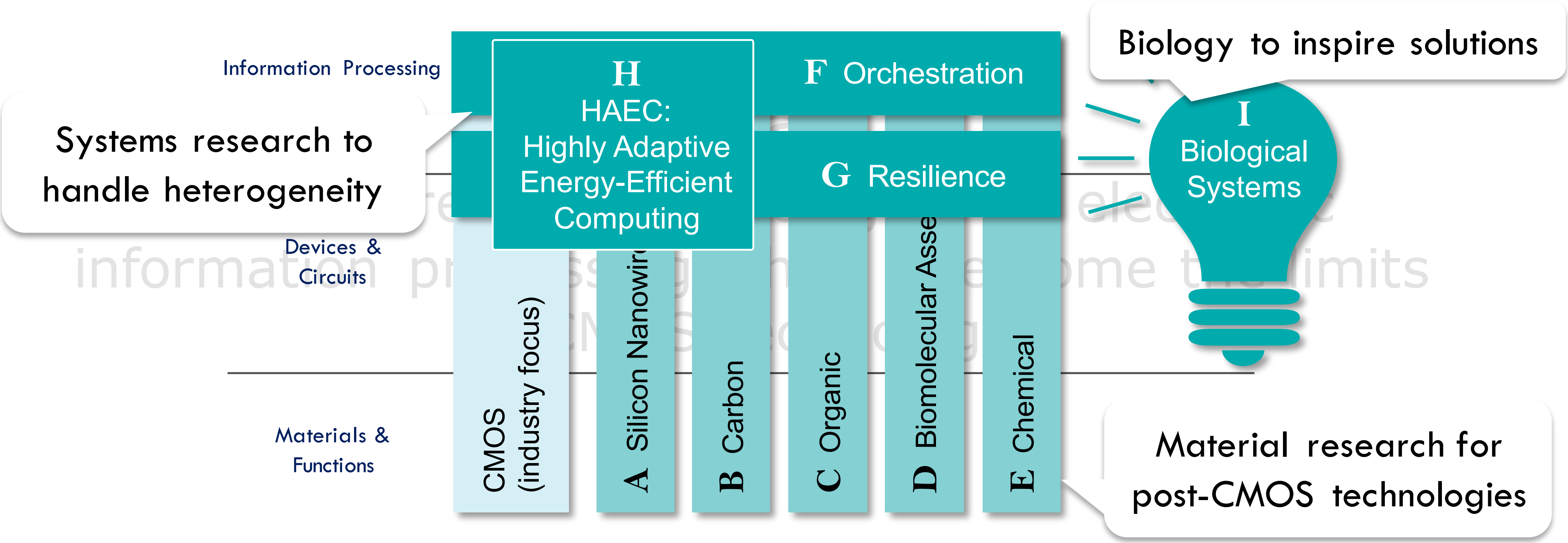
jeronimo.castrillon@tu-dresden.de

Cfaed: Research program (revisited)

Goal

to explore new technologies for electronic information processing which overcome the limits of CMOS technology

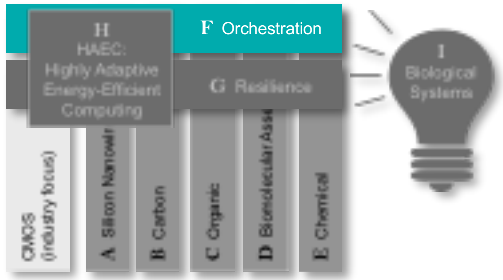
Cfaed: Research program (revisited)



Orchestration Path

Mission

Turn materials breakthroughs into application performance



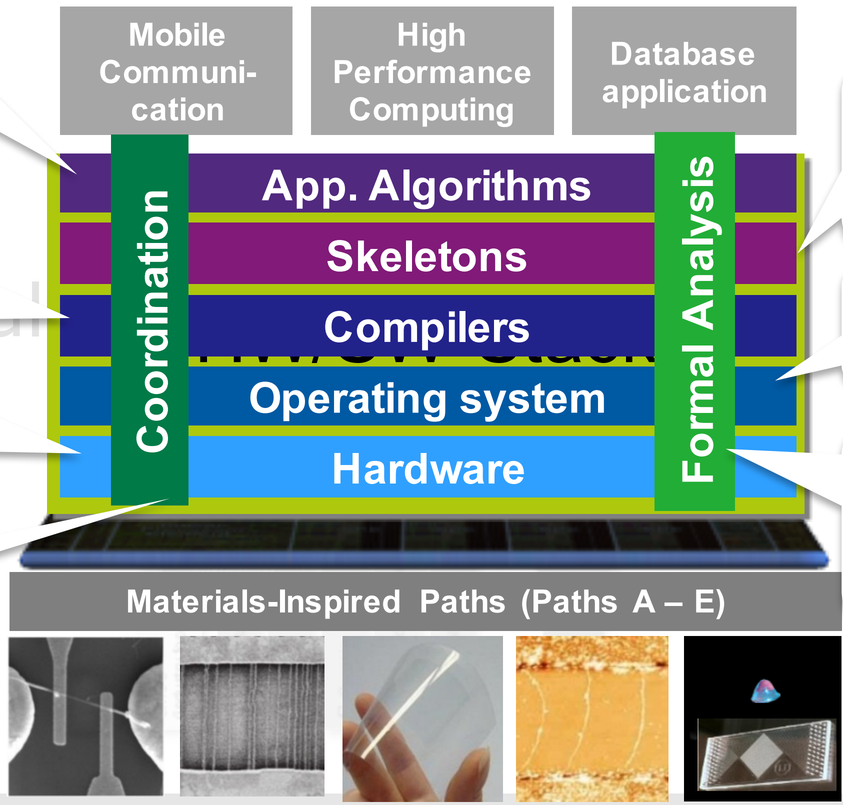
Orchestration Path

Underlying algorithms: self-adjusting

Code synthesis: variant generation

Heterogeneous “template”

Cross-layer strategies



Programming abstractions

“Micro-kernels”

System analysis: modeling, verification & reasoning

Orchestration Path: the team

- Investigators
  - Prof. Uwe Aßmann
  - Prof. Franz Baader
  - Prof. Christel Baier
  - Prof. Jeronimo Castrillon
  - Prof. Gerhard Fettweis
  - Prof. Christof Fetzer
  - Prof. Jochen Fröhlich
  - Prof. Hermann Härtig
  - Prof. Wolfgang Lehner
  - Prof. Wolfgang E. Nagel
  - Dr. Marcus Völp
  - Prof. Axel Voigt

Orchestration Path: the team (2)

- PhD/Postdocs
  - Nils Asmußen
  - Andres Goens
  - Sebastian Haas
  - Dirk Habich
  - Immo Huismann
  - Tomas Karnagel
  - Sven Karol
  - Sascha Klüppelholz
  - Linda Leuschner
  - Matthias Lieber
  - Siqi Ling
  - Steffen Märker
  - Johannes Mey
  - Benedikt Nöthen
  - Michael Raitza
  - Norman Rink
  - Annett Ungethüm

Outline

- Introduction
- **HW/SW stacks**
- Orchestration stack
- Outlook
- Concluding remarks

What is a stack

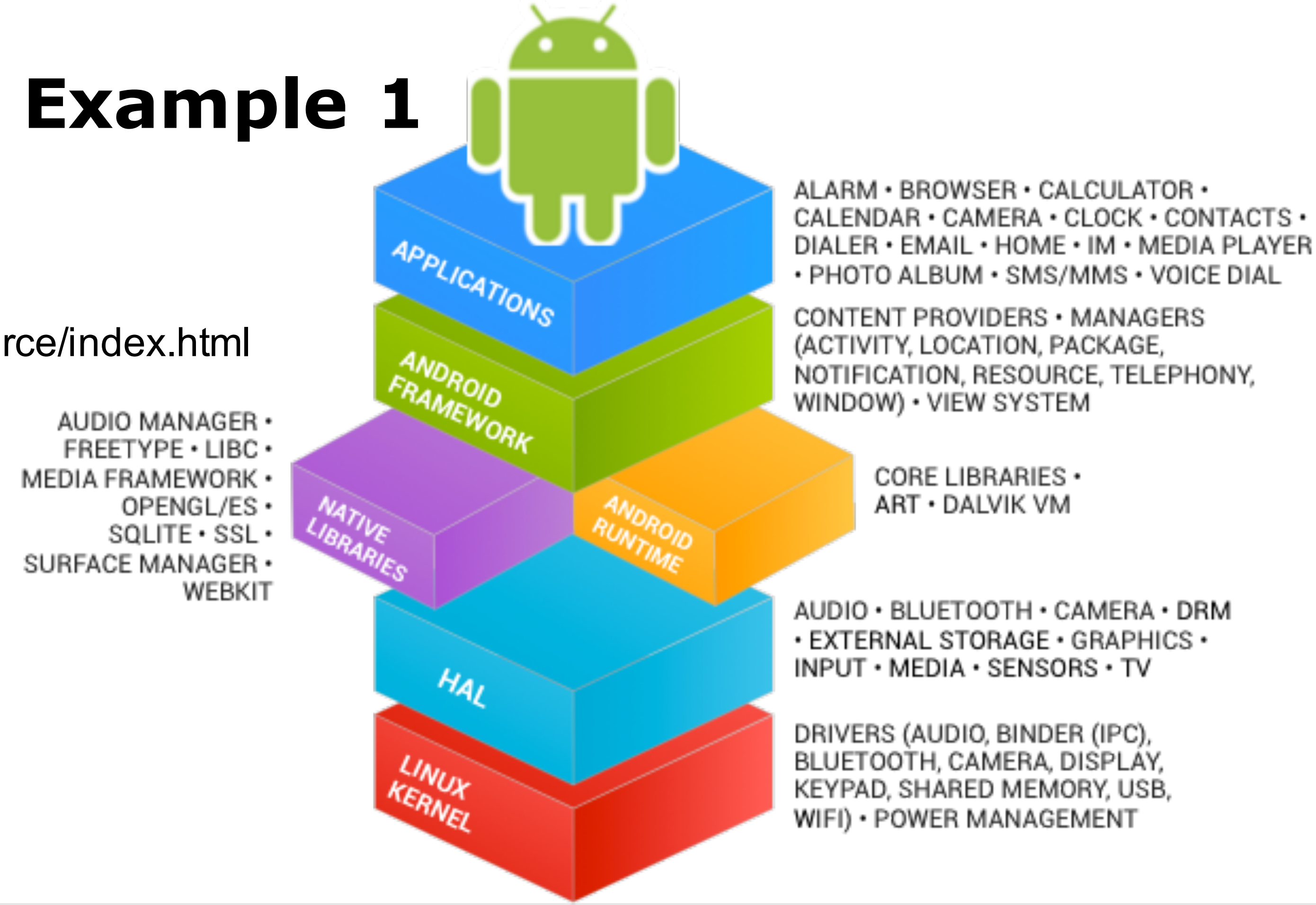
- Software components that form a complete platform (where you can run applications on)
  - Web app: Operating system, web server, database, programming language

- Provide means to handle complexity

HW/SW Stack: Example 1

Android sack

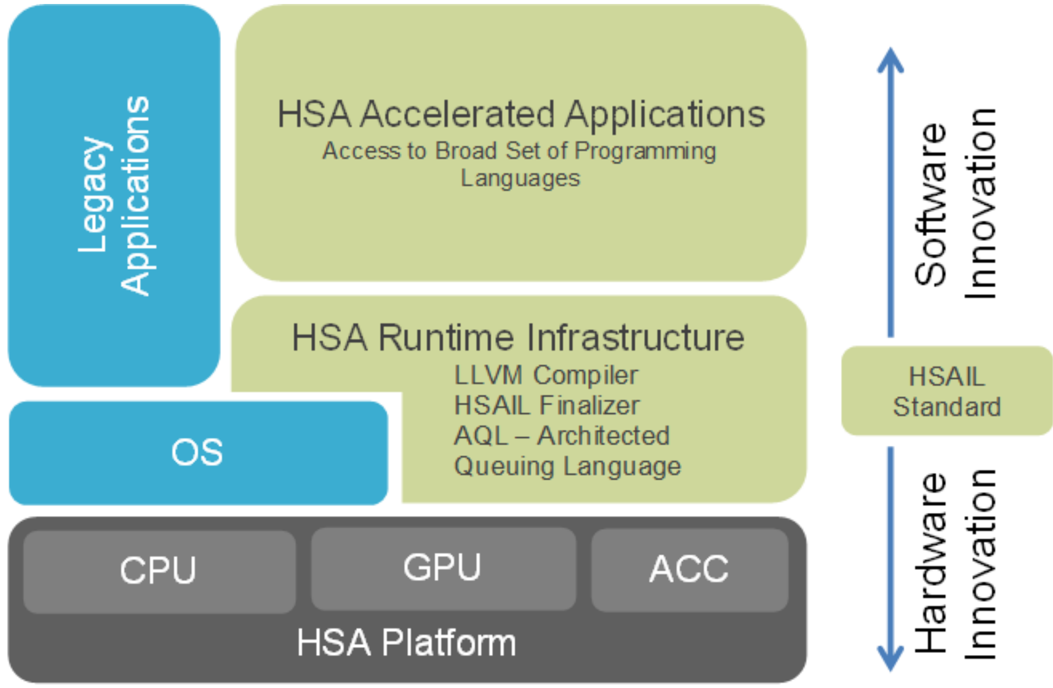
Source:
https://source.android.com/source/index.html



HW/SW Stack: Example 2

HSA Stack

Source: Heterogeneous System Architecture
http://developer.amd.com/resources/heterogeneous-computing/what-is-heterogeneous-system-architecture-hsa/



HSA Solution Stack

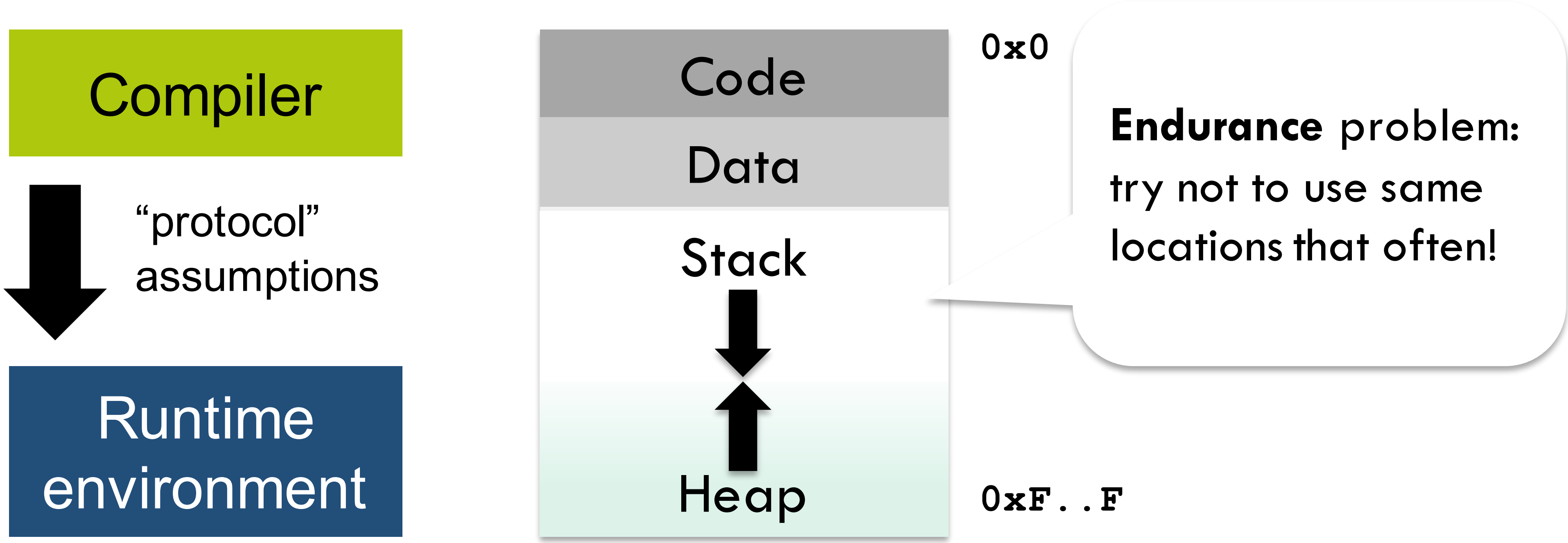
Stacks and wild heterogeneity

- Abstract/hide complexity
- Leave as much visible as needed for efficiency

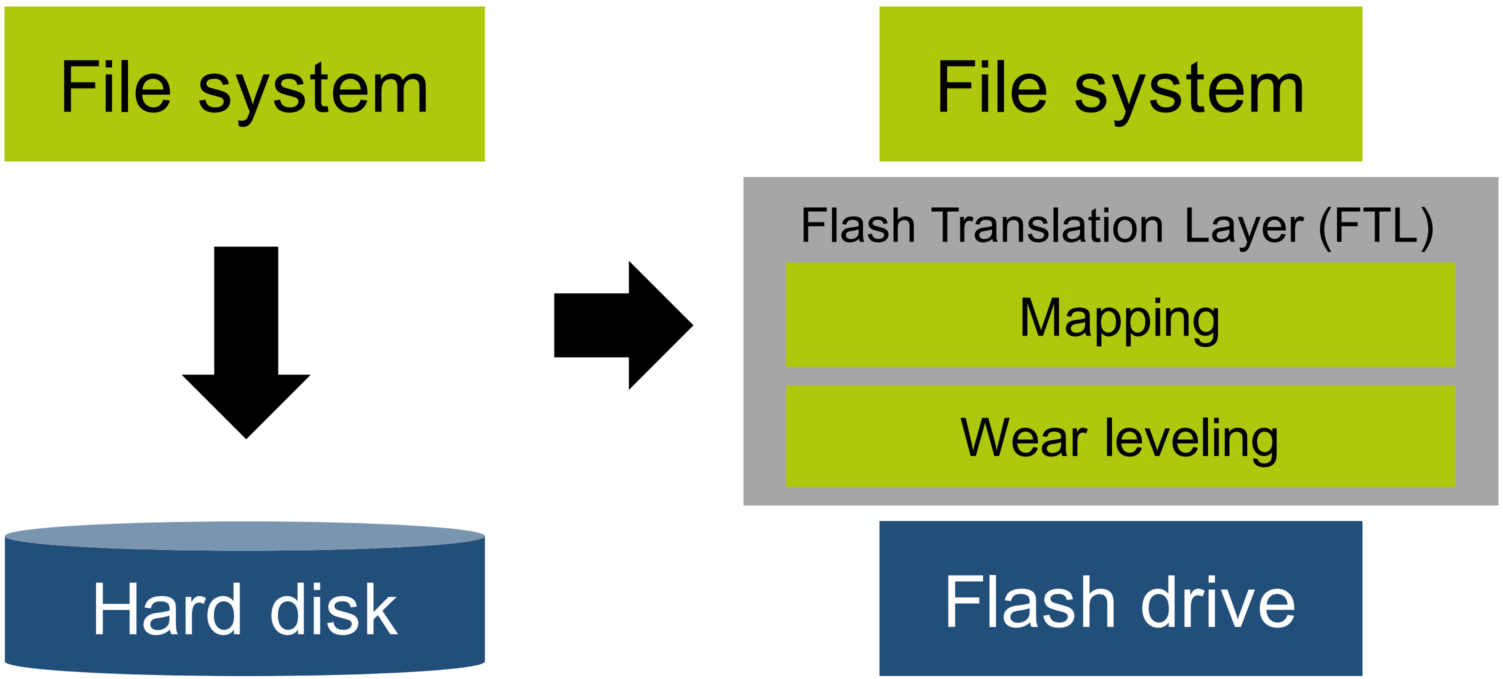
- Emerging techs.: rethink established layers
  - Languages, compilers, operating systems, …

HSA Solution Stack

Example: Compiler – Runtime



Example: Flash FTL

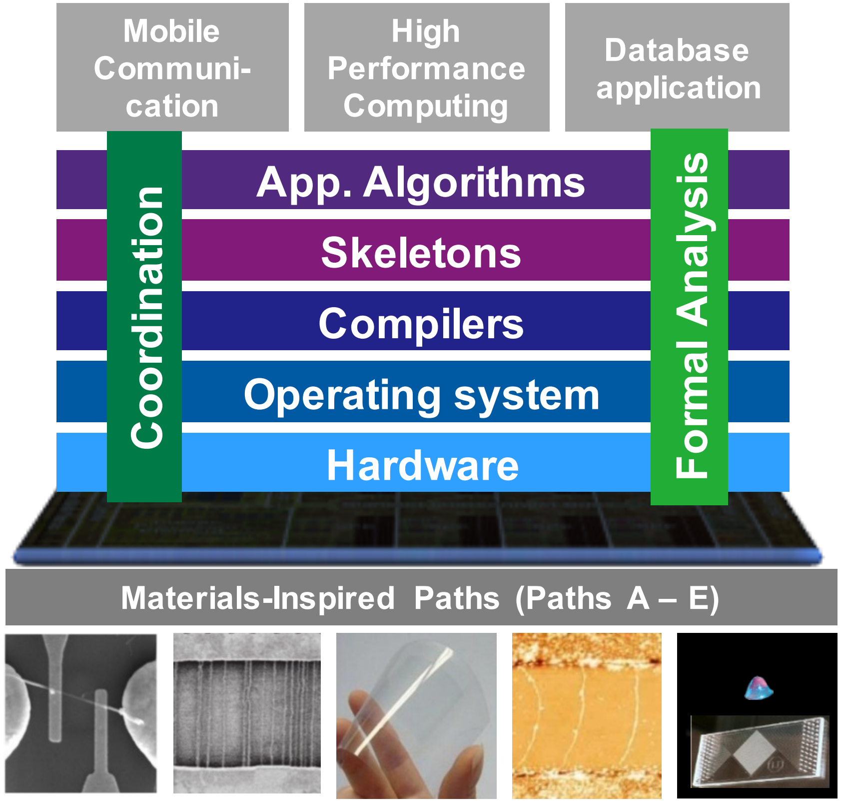


Outline

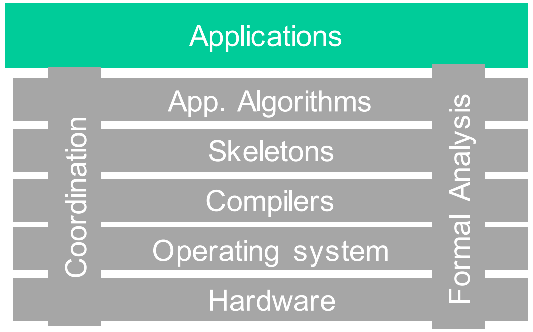
- Introduction
- HW/SW stacks
- **Orchestration stack**
- Outlook
- Concluding remarks

Recall: orchestration stack

- Heterogeneous materials
  - In Cfaed: Along with orchestration (see other DMA presentations)
- Research plan
  - Start with heterogeneous CMOS
  - Tiled for scalability, integrability
  - Pilot projects with materials as they become available



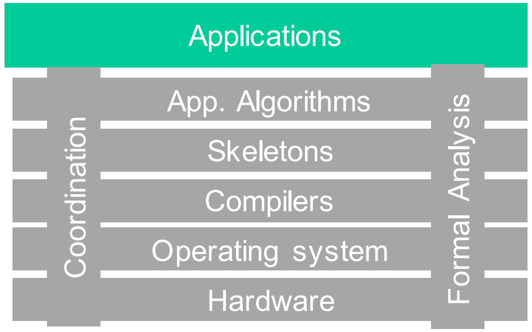
TECHNISCHE UNIVERSITÄT DRESDEN · cfaed · GLOBALFOUNDRIES · X-FAB



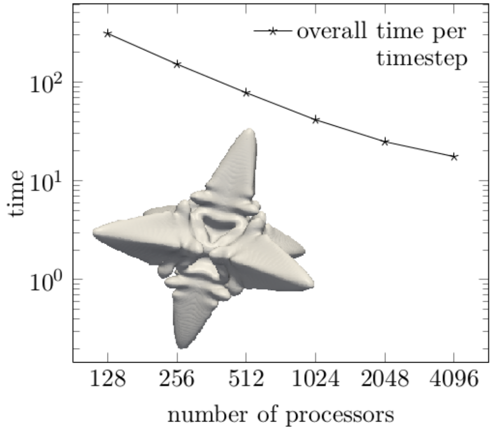
Applications/Drivers

- Mobile communication: typically run on heterogeneous
- Data-bases: benefits of heterogeneity (see below)
- HPC: high level languages, abstractions for scalability, productivity and portability
  - Adaptive Finite Element Methods (FEM)
  - Computational Fluid Dynamics (CFD)

Applications/Drivers (2)

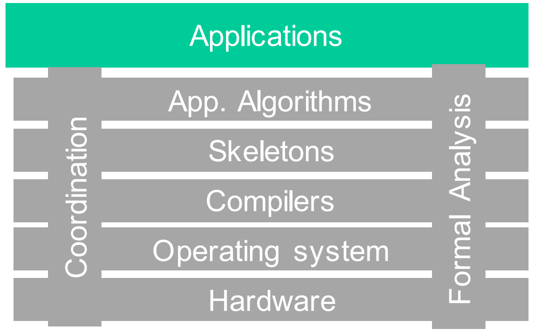


- Adaptive Finite Element Methods (FEM)
  - Working with multiple-meshes
  - Communication across subdomains
  - Requires repartitioning ➔ load balancing
  - Working on
    - Portable *templatized* library
    - First results on heterogeneous platforms



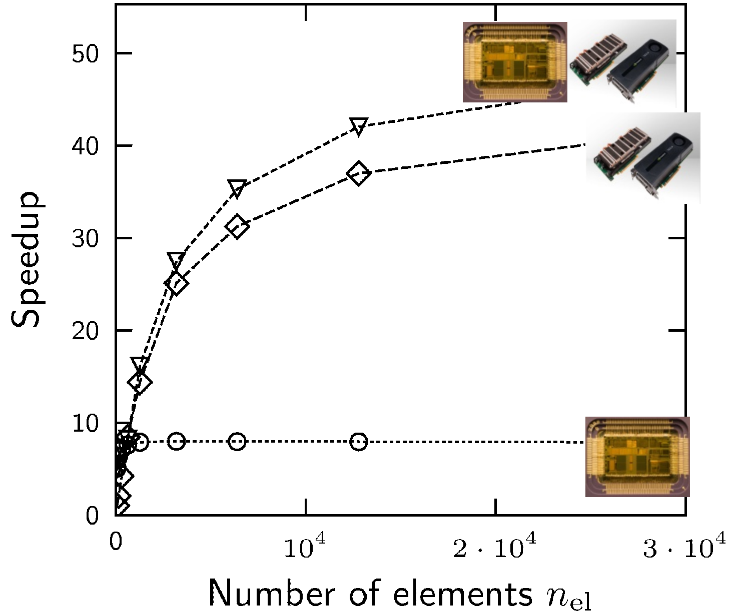
Dendritic growth

[Witkowski15]

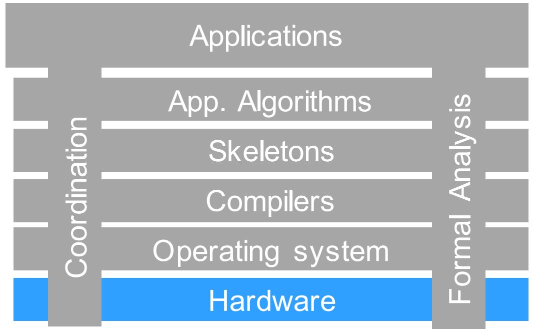


Applications/Drivers (3)

- Computational Fluid Dynamics (CFD)
  - Large applications (10^{10} unknowns)
  - Communication/computation is key
  - Working on
    - Heterogeneous implementation
    - More versatile fundamental concepts

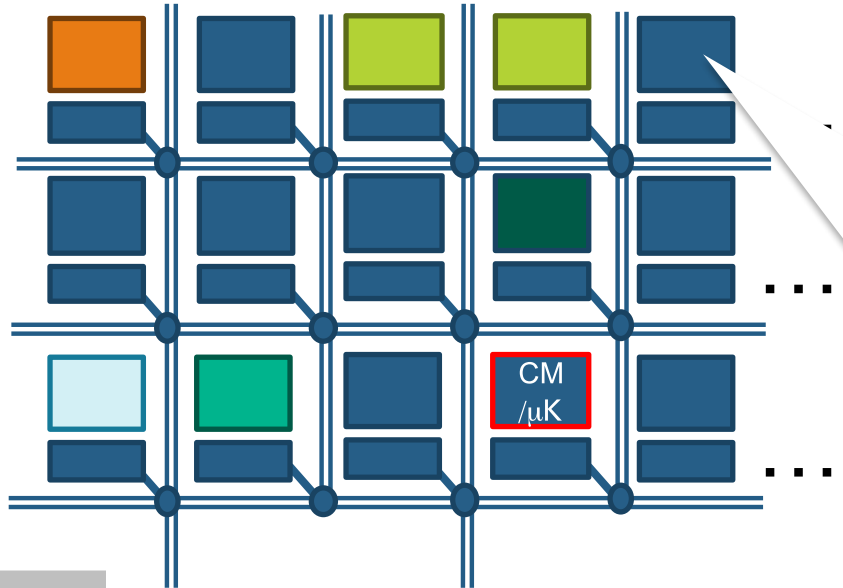


[Huismann15]



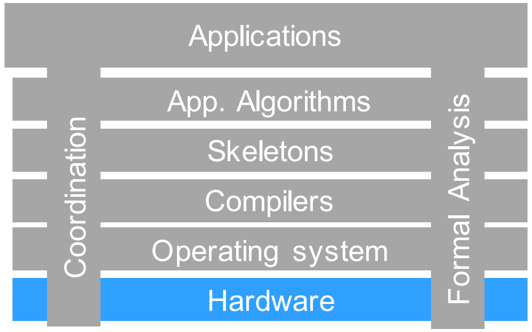
Hardware

- Tomahawk: heterogeneous CMOS



- Tiled (with NoC) for scalability
- Allows to integrate wildly heterogeneous components
- CoreManager (CM): HW support for fine-grained task dispatching

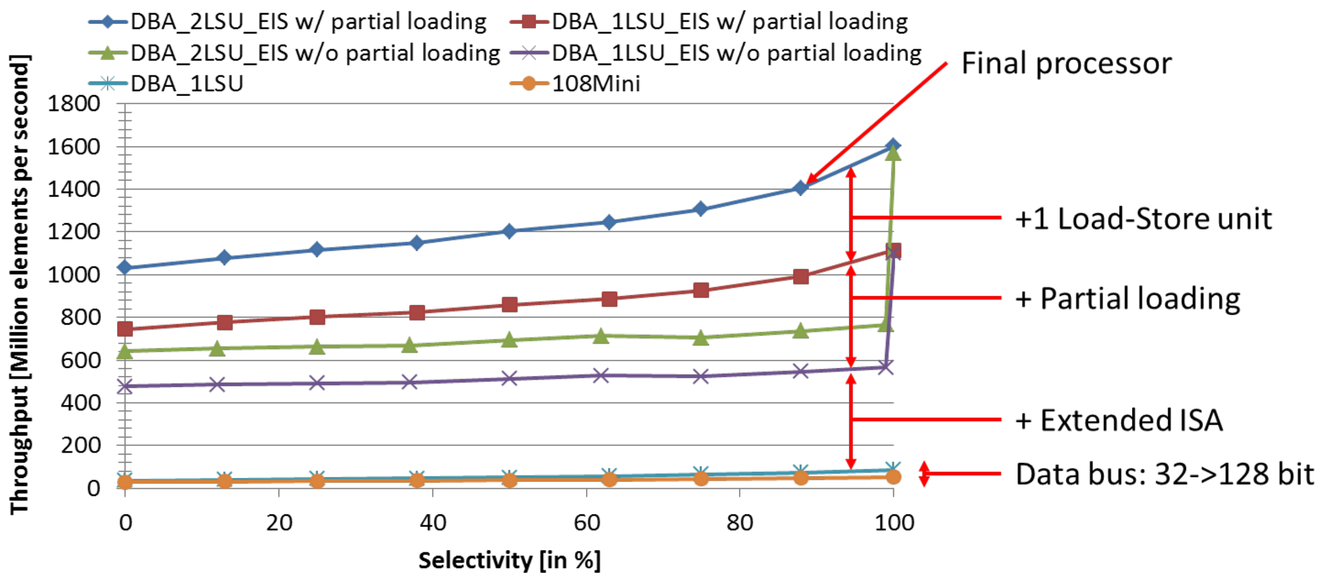
[Arnold13]



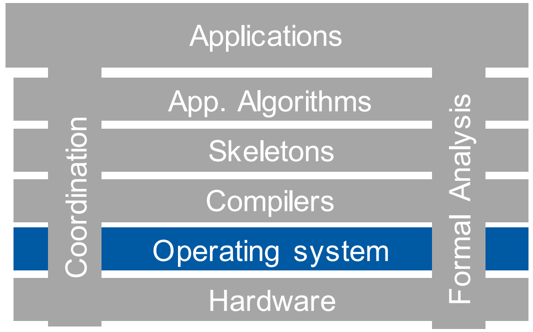
Hardware

- DB-processor for sorting algorithms

Selectivity:

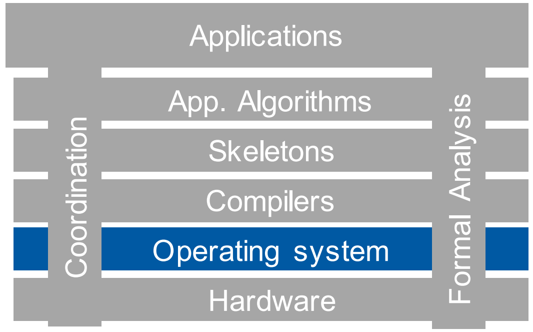


[Arnold14]



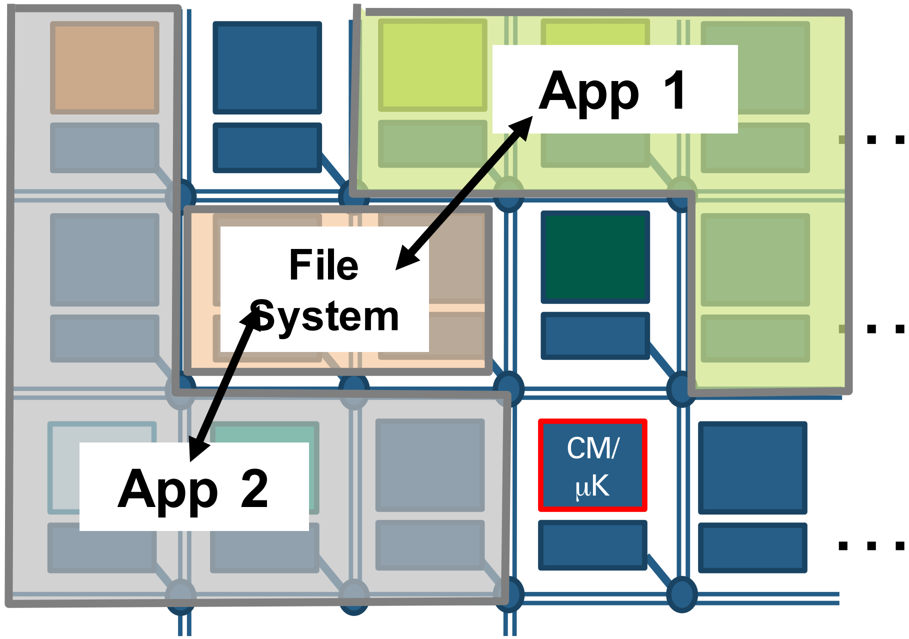
Operating Systems

- Microkernel for heterogeneity
  - Isolation (at NoC level)
    - Dynamic compartments
    - Isolation at network boundary
  - Remote control: no need for OS everywhere
    - Simplify integration of novel materials
    - OS services on remote cores



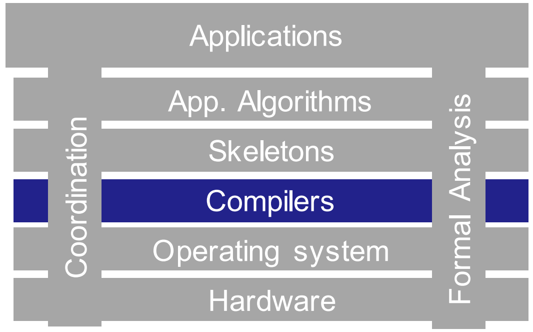
Operating Systems

- Microkernel for heterogeneity



Isolation & remote control

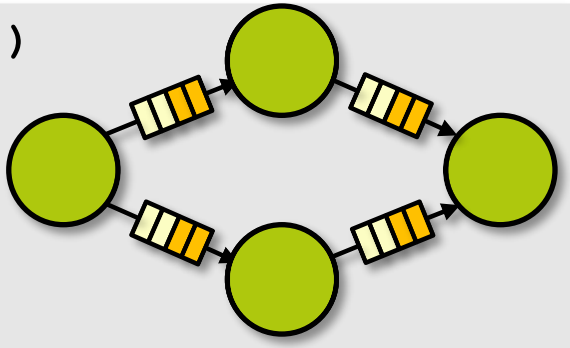
[Asmussen15]



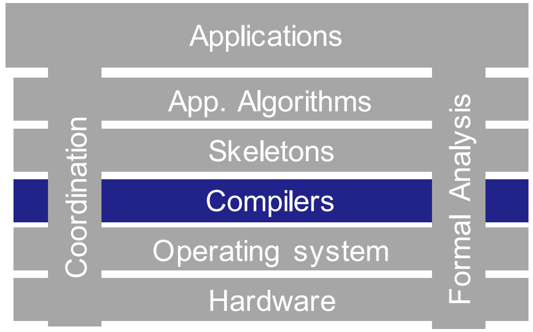
Compilers

- Parallel dataflow programming languages and compilers

```
__PNkpn AudioAmp __PNin(short A[2]) __PNout(short B[2])
                 __PNparam(short boost){
  while (1)
    __PNin(A) __PNout(B) {
      for (int i = 0; i < 2; i++)
        B[i] = A[i]*boost;
    }}
__PNprocess Amp1 = AudioAmp __PNin(C) __PNout(F) __PNparam(3);
__PNprocess Amp2 = AudioAmp __PNin(D) __PNout(G) __PNparam(10);
```

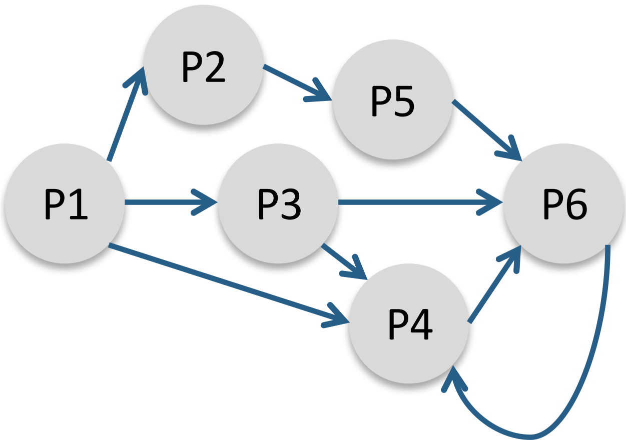


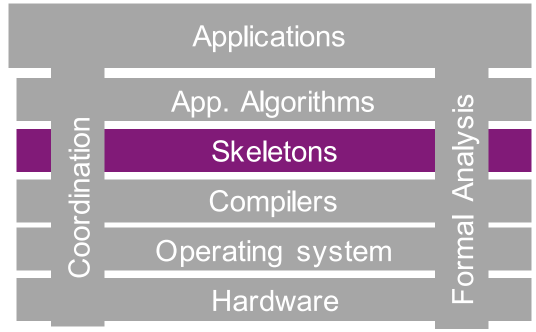
[Castrillon14]



Compilers

- SW synthesis for Tomahawk
  - Static/dynamic mapping
  - Automatic code generation

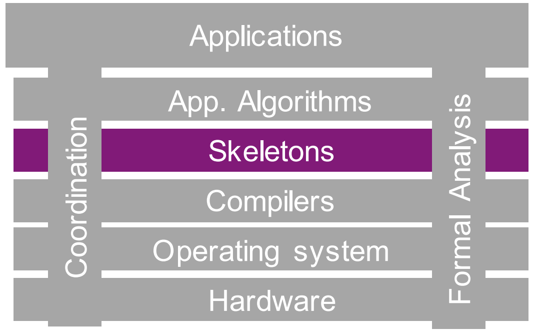




Skeletons

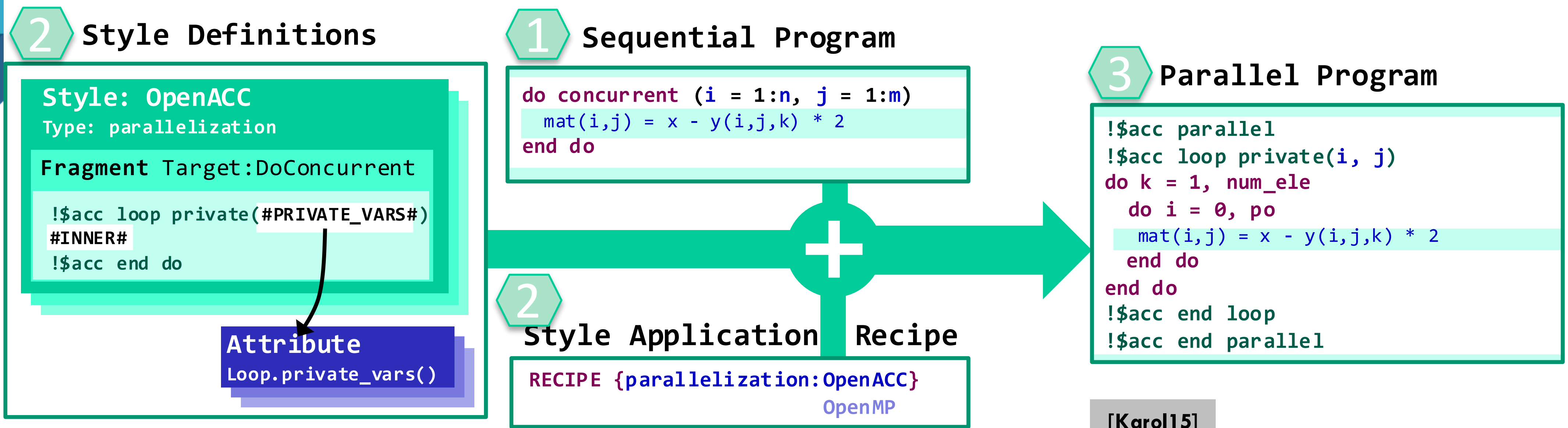
- Basic building blocks for parallelism
- "Heterogeneous" programming
  - Different possible implementations of same principles (CUDA, Fortran, …)
  - Advanced "pre-compiler" to generate and manage variants

  ➔ **Productivity**

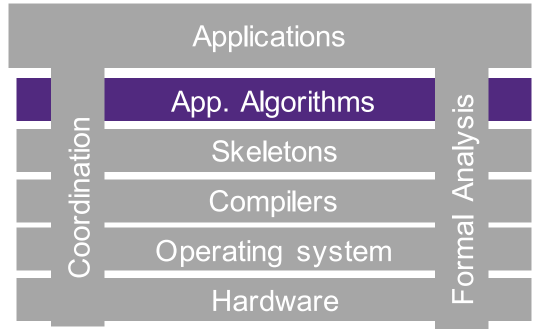


Skeletons

- Basic building blocks for parallelism

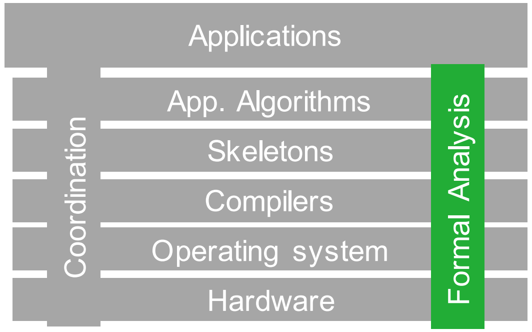


[Karol15]



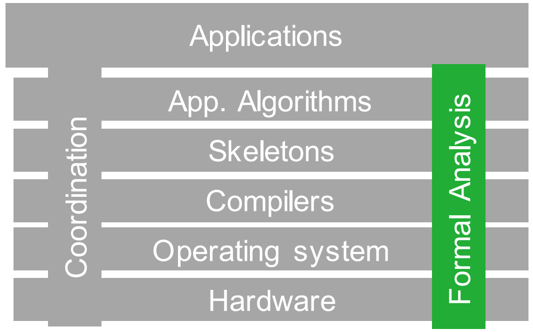
Algorithms

- Raise the level of abstraction
  - High-level **domain specific languages** (more later)
  - Algorithmic transformations: more optimization room
  - Algorithmic specification: allow for adaptability



Formal analysis

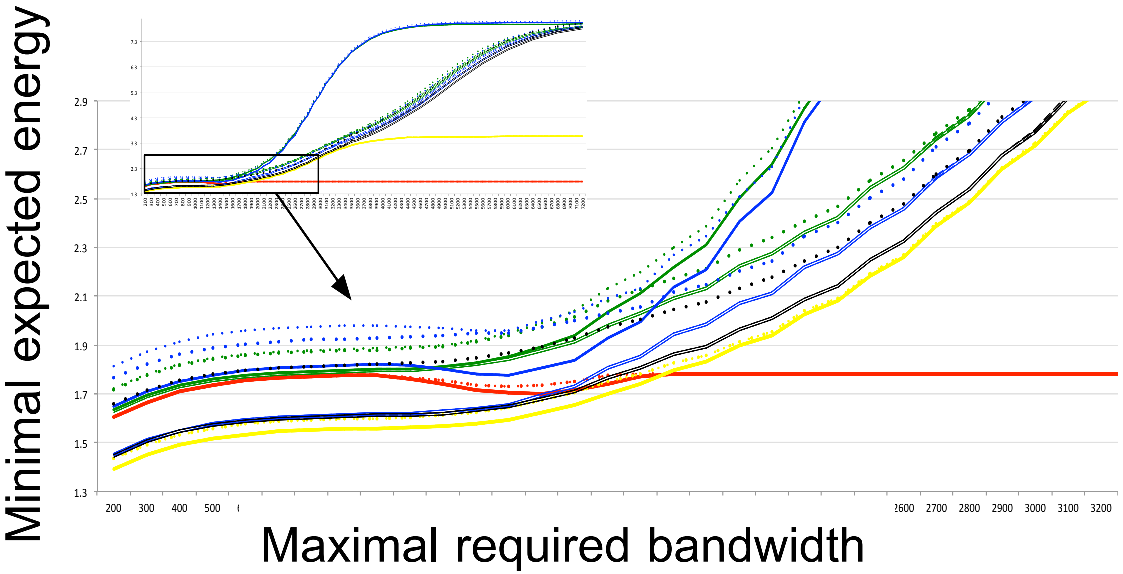
- Probabilistic model checking (PMC)
  - **Analyze** approaches across layers of the stack
  - Work on models for heterogeneous systems
  - Examples:
    - Probability of finishing N tasks in T time
    - Probability of finishing N tasks with energy budget B
    - Trade-off analysis cost-utility (energy vs. performance)
  - Model protocols: Spinlock, barriers, eBond, …



Formal analysis

- Probabilistic model checking (PMC)

eBond



[Baier14]

Outline

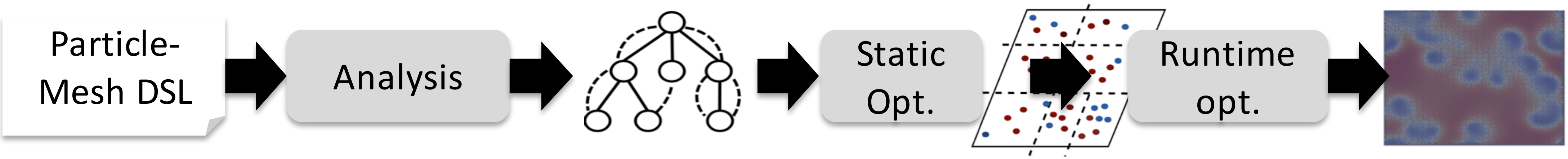
- Introduction
- HW/SW stacks
- Orchestration stack
- **Outlook**
- Concluding remarks

Current and future work

- Further work on languages and methodologies
  - Computational biology
  - Chemical information processing
- Technology analysis at system-level
  - Pros (USPs) and cons
  - Attempt skip incremental improvement
  - System-level benefit based on assumptions

Language for computational biology

- Particle and mesh-based Simulations
  - Discrete models of physical phenomena
- Goal: Abstract programming language + optimization for heterogeneous HPC-clusters



[Karol15]

Language for computational biology

```
client grayscott
  integer, dimension
  real(..), dimensio
  integer

  add_arg(k_rate,<#r
  add_arg(F,<#real(m
  add_arg(D_u,<#real
  add_arg(D_v,<#real

  ppm_init(1)

  U = create_field(1
  V = create_field(1
  topo = create_topo
  c = create_particl

  allocate(displace(
  call random_number
  call c%move(displa
  call c%apply_bc(in

  global_mapping(c,
  discretize(U,c)
  discretize(V,c)
  ghost_mapping(c)

  foreach p in parti
    U_p = 1.0_mk
    V_p = 0.0_mk
    if (((x_p(1)-0.5

      U_p = 0.5_mk
      V_p = 0.25_mk
    end if
  end foreach
```

ArithmeticOperation × | typeof_BinaryExpression × | typeof_AssignmentExpression ×

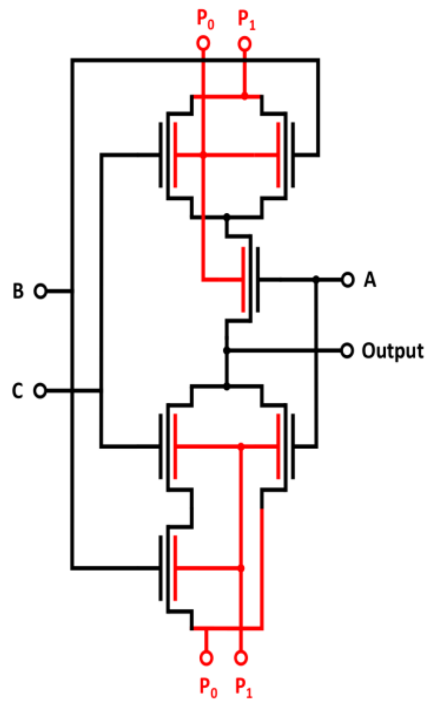
```
import LaplacePSE2D from stencils as Lap
import BcdefPeriodic from topologies as bcdef
import RungeKutta from schemes as rk4

module GrayScott
  external real kRate = 1.0;   "reaction rate"
  external real F;             "reaction parameter"
  external real Du = 1.0;      "diffusion constant of U"
  external real Dv = 1.0;      "diffusion constant of V"

  phase initialize
    << ... >>
  end initialize

  phase solve
    field <real , 1> U;
    field <real , 1> V;
    integer t;

    δU/δt = Du * ∇² U − U * V² + F * (1 − U);
    δV/δt = Dv * ∇² V + U * V² − (F + kRate) * V;
  end solve

  phase finalize
    << ... >>
```

```
_mk  * c%h_avg#>)
.0_mk, 1.0_mk], "Lap")

am_op_dcpse,
der=>2,c=>1.0_mk])

tt_rhs,[U=>c,V],  rk4)

h sca_fields(U,V,dU,dV)
 + F*(1.0_mk-U_p)
 - (F+k_rate)*V_p
```

[Karol15]

SiNW: Components and reconfigurability

- Edge of SiNW over other technologies
  - P/N reconfigurability
  - Multi-gate devices (with low latency penalty)
- Components based on assumptions
  - New uArchitectures?
  - New pipeline structures and compiler optimizations?

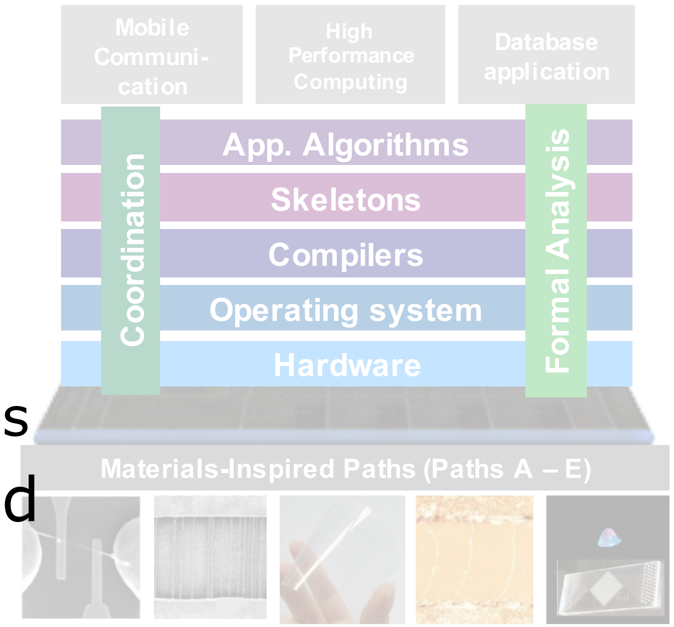


Outline

- Introduction
- HW/SW stacks
- Orchestration stack
- Outlook
- **Concluding remarks**

Summary

- Orchestration path: Multi-disciplinary effort to turn **materials breakthroughs** into **application performance**
- HW/SW stack to handle heterogeneity
  - Using CMOS as vehicle
  - Prepare for "*wildly*" heterogeneous systems
- Interact with material experts, understand technologies and analyze at system-level



Thanks!

Retreat: Orchestration and resilience paths (03.2015)

References

[Witkowski15] Witkowski, T., Ling, S., Praetorius, S., & Voigt, A. (2015). Software concepts and numerical algorithms for a scalable adaptive parallel finite element method. Advances in Computational Mathematics, 1-33

[Huismann15] Huismann, I., Stiller, J., & Fröhlich, J. (2015). Two-level parallelization of a fluid mechanics algorithm exploiting hardware heterogenity. Computers & Fluids

[Arnold13] Arnold, O., Matus, E., Noethen, B., Winter, M., Limberg, T., & Fettweis, G. (2014). Tomahawk: Parallelism and heterogeneity in communications signal processing MPSoCs. ACM Transactions on Embedded Computing Systems (TECS), 13(3s), 107

[Arnold14] Arnold, O., Haas, S., Fettweis, G., Schlegel, B., Kissinger, T., Karnagel, T., & Lehner, W. (2014). HASHI: An Application-Specific Instruction Set Extension for Hashing. ADMS@ VLDB, 25-33

[Asmussen15] Asmussen, N., Volp, M., Nothen, B., & Ungethum, A. (2015, April). Demo abstract: Taming many heterogeneous cores. In Real-Time and Embedded Technology and Applications Symposium (RTAS), 2015 IEEE (pp. 329-329)

[Castrillon14] J. Castrillon and R. Leupers, Programming Heterogeneous MPSoCs: Tool Flows to Close the Software Productivity Gap. Springer, 2014

[Karol15] Sven Karol, Well-Formed and Scalable Invasive Software Composition. PhD thesis. May 2015

[Baier14] Baier, C., Dubslaff, C., & Klüppelholz, S. (2014, July). Trade-off analysis meets probabilistic model checking. In Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)

[Karol15] S. Karol, P. Incardona, Y. Afshar, I. Sbalzarini, and J. Castrillon. Towards a Next-Generation Parallel Particle-Mesh Language, in Domain-Specific Language Design and Implementation (DSLDI). July 2015