CHAIR FOR COMPILER CONSTRUCTION

Dataflow programming for heterogeneous computing systems

Jeronimo Castrillon
Cfaed Chair for Compiler Construction
TU Dresden
jeronimo.castrillon@tu-dresden.de

Tutorial: Algorithmic specification, tools and algorithms for programming heterogeneous platforms. PACT Conference. San Francisco, October 19th 2015

Outline

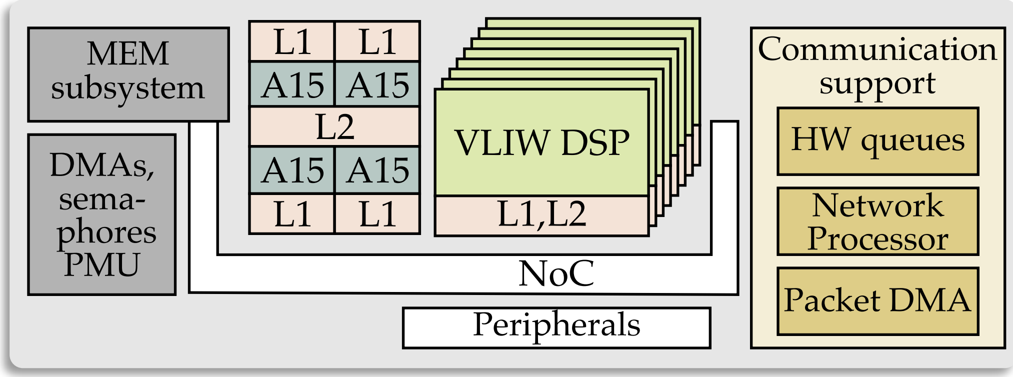
- Heterogeneous systems
- Dataflow models
- Analysis and synthesis
- Summary

Outline

- **<u>Heterogeneous systems</u>**
- Dataflow models
- Analysis and synthesis
- Summary

Heterogeneous computing systems

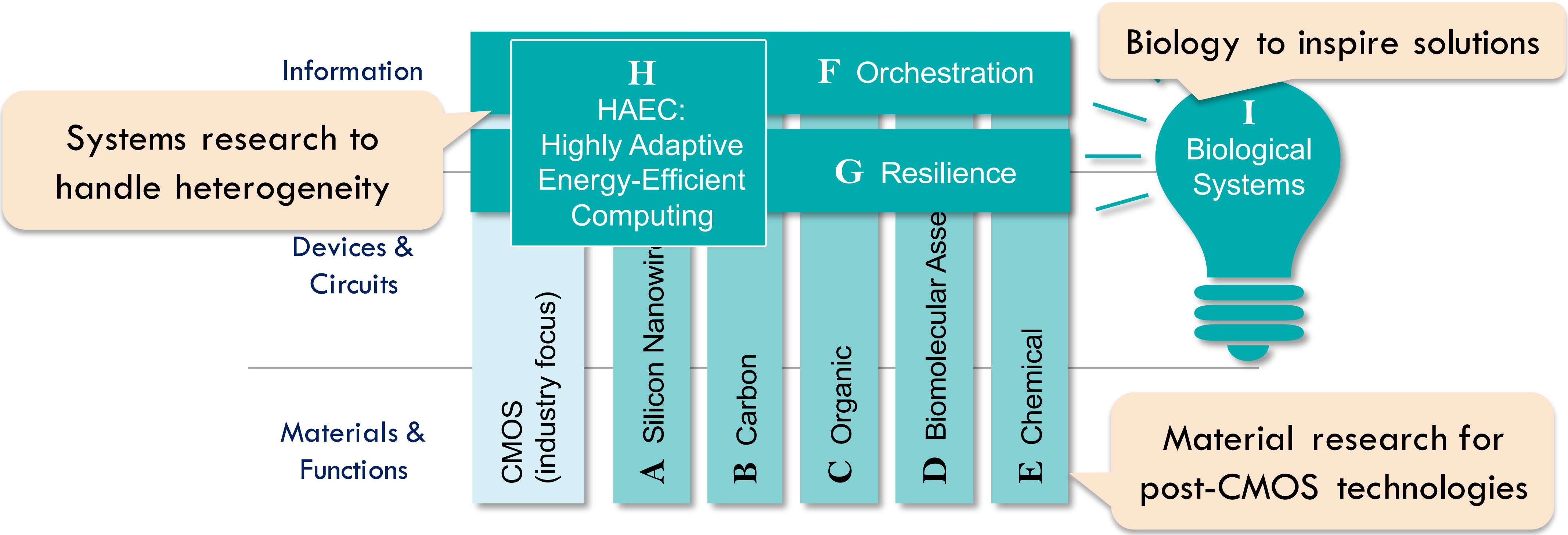
- Today's heterogeneity
  - Desktop/HPC: GPGPU, GP + ACC
  - Embedded: RISCs, DSPs, ASIPs, …
  - → Resources: different performance/energy characteristics



- Tomorrow's heterogeneity: Emerging technologies
  - Heterogeneity beyond performance and energy
    - Reliability/error tolerance
    - Computation model

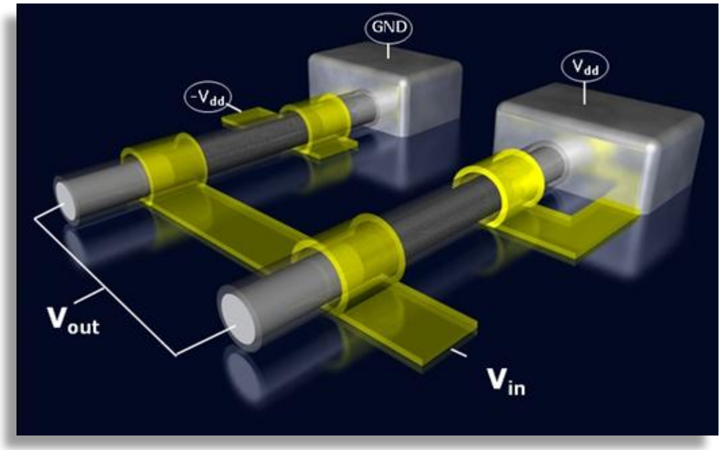
Heterogeneity: Cfaed Vision

- German Excellence Cluster: Goal – "**to explore new technologies for electronic information processing which overcome the limits of CMOS technology**"

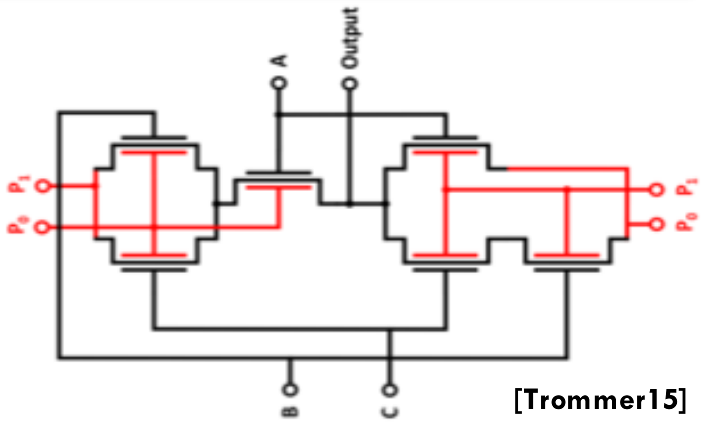


Heterogeneity: Example SiNW

- SiNW: Silicon Nanowires
- Multi-gate devices with less performance penalty
- Reconfigurable P/N functionality



- Possibilities
  - New micro architectures
  - New pipeline structures
  - New field programmable devices

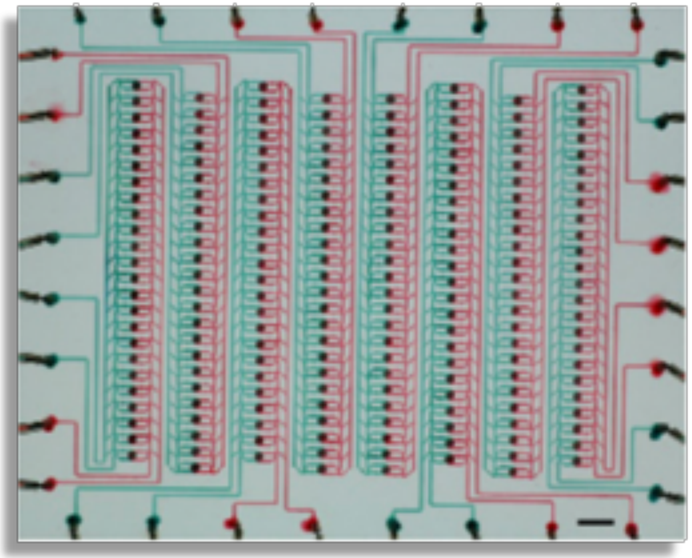


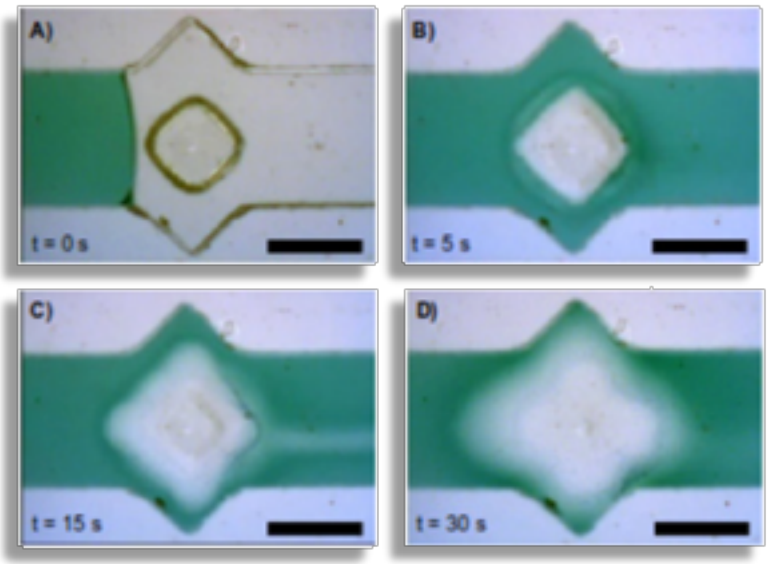
[Trommer15]

Heterogeneity: Example chemical processing

- Lab-on-Chips: for sensing, analysis and test
- Also for computing?
  - Different kinds of transistors
  - Oscillators and other components

Fundamentally different way of computation

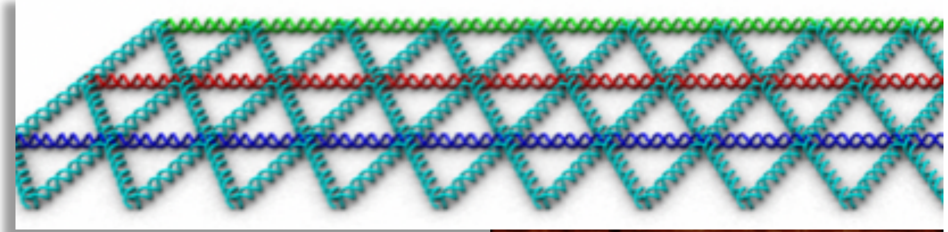


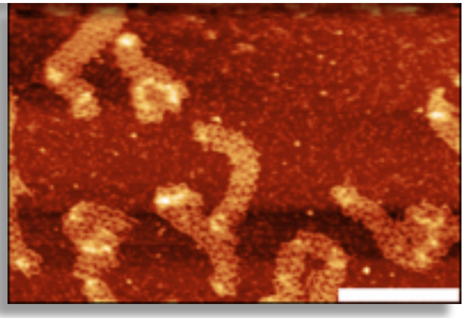


[Voigt14]

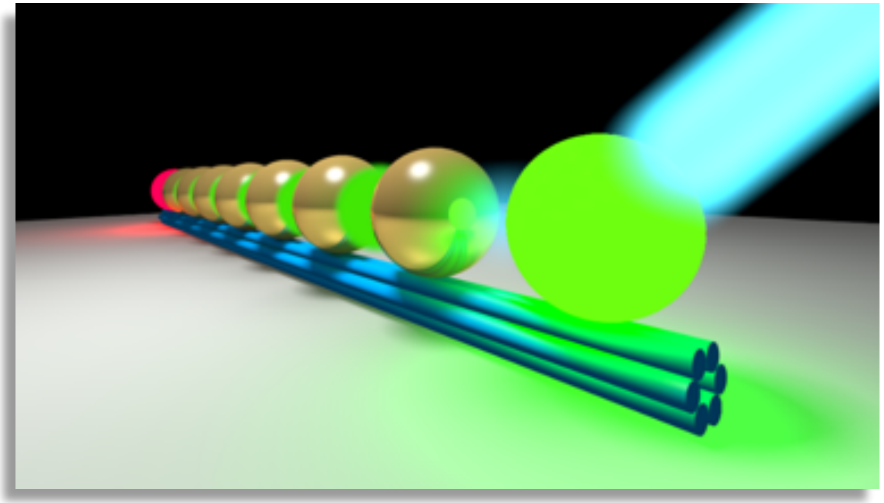
Heterogeneity: Example DNA origami

- DNA origami: Self-assembled 2D/3D structures made of DNA strands



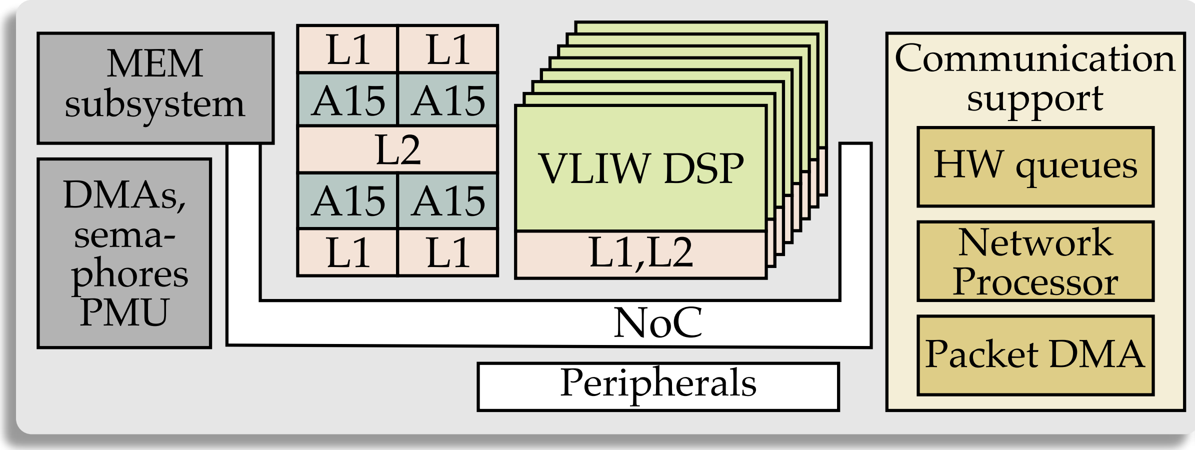


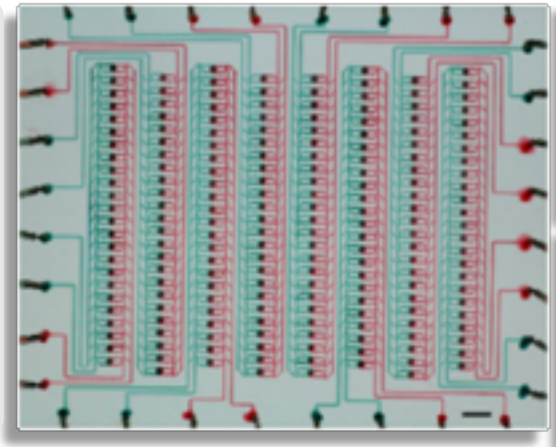
- Use structures to build advance electronic devices
- Example: Plasmonic waveguides

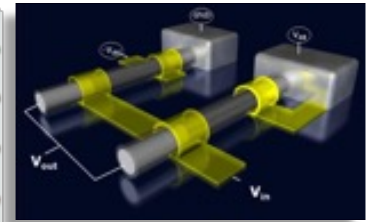


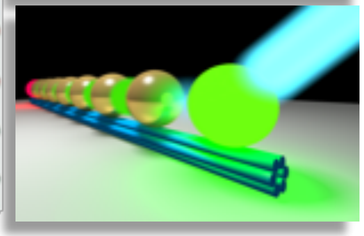
Courtesy: Thorsten Lars-Schmidt
https://cfaed.tu-dresden.de/schmidt-home

Consequence of heterogeneity









Already difficult to program them today, what about tomorrow?

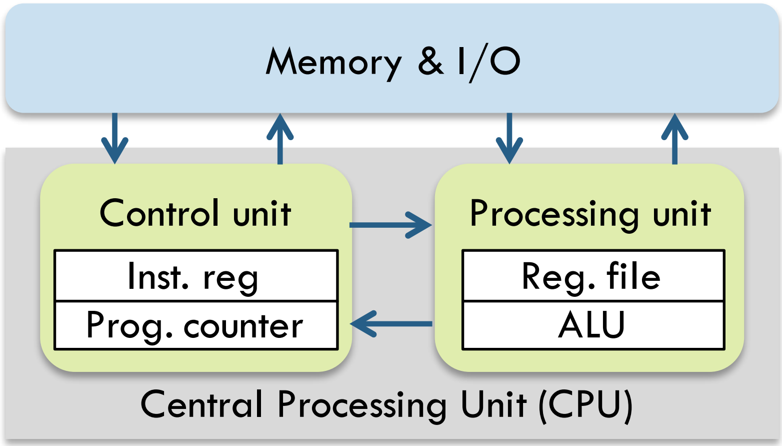
Need for models and abstraction

Outline

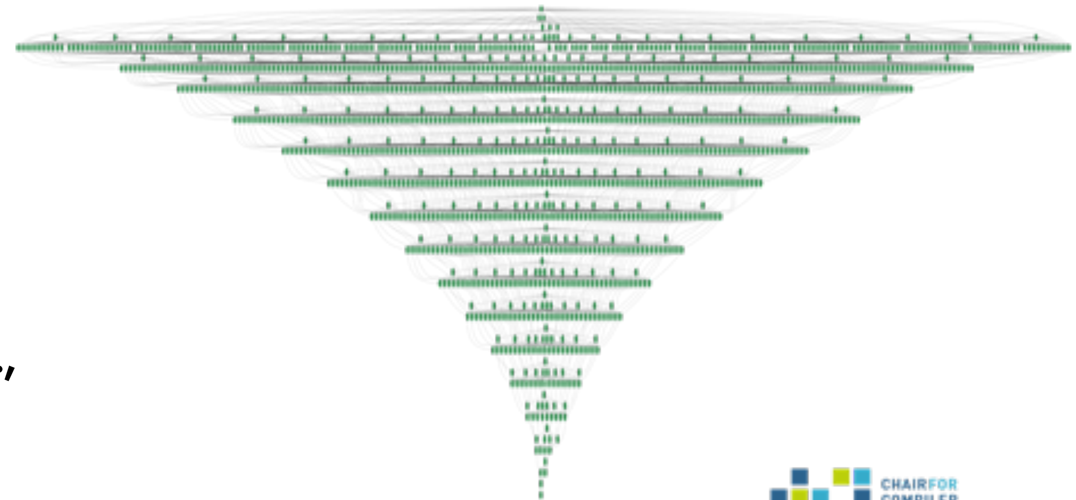
- Heterogeneous systems
- **Dataflow models**
- Analysis and synthesis
- Summary

Models: Introduction

- Von Neumann model makes things complicated
  - Sharing state
  - Data races

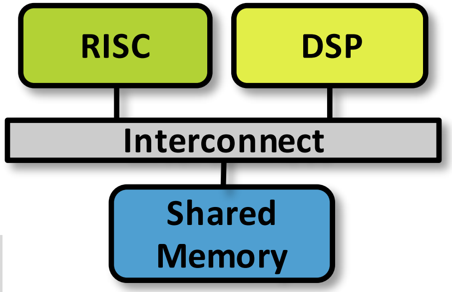


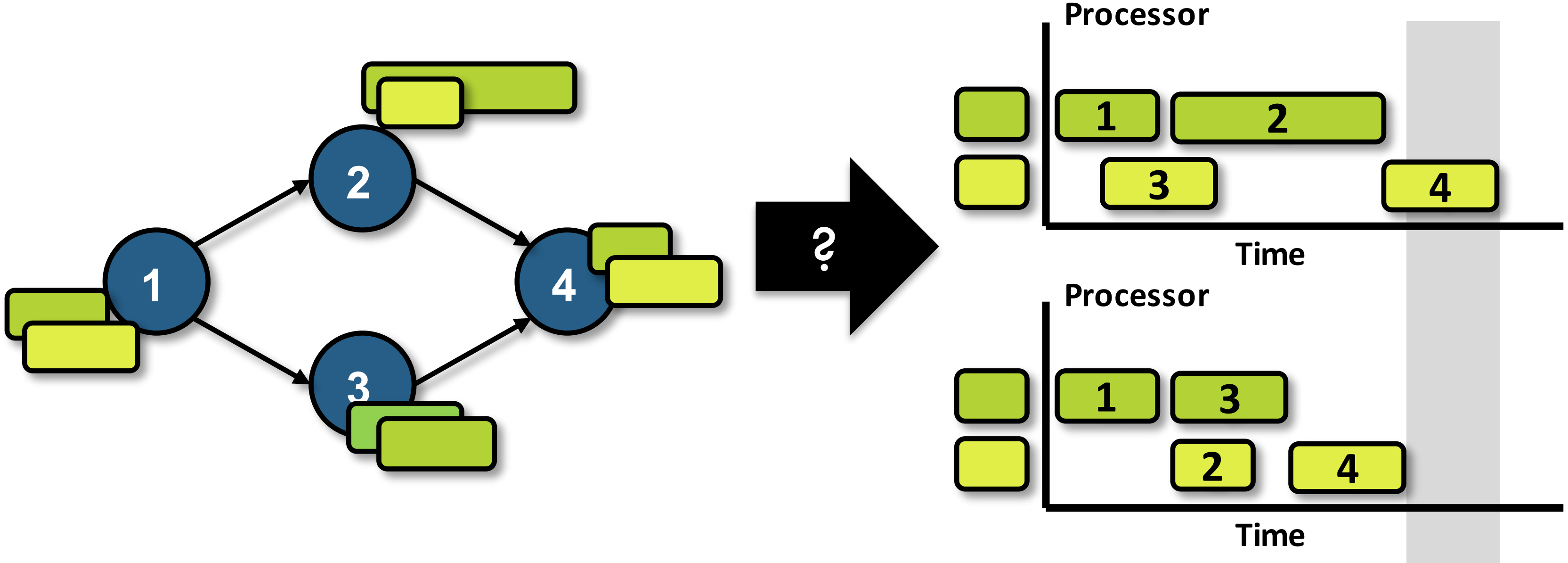
- Task graphs: A simple parallel programming model
  - Intel TBB, .NET Task parallel library (TPL), OpenMP Tasks, …
  - Runtime and data management, e.g., StarPU [Augonnet10]



Directed acyclic task graphs

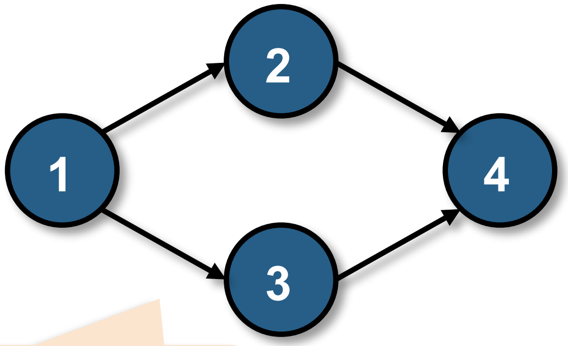
- Very well studied, see for example [Kwok99]
- *Difficult* problem for heterogeneous systems

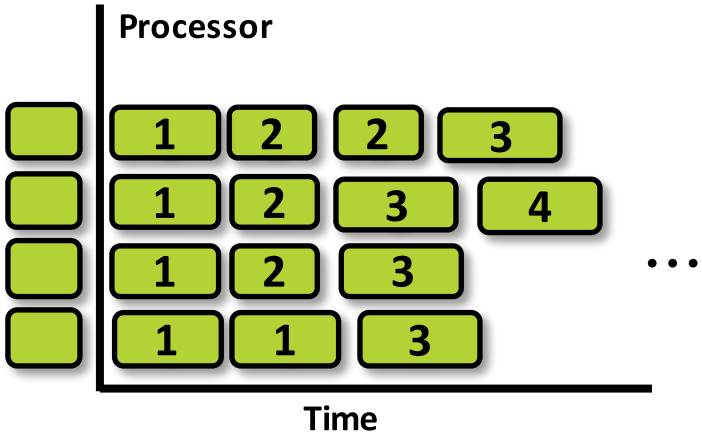




Dataflow models

- Also a graph representation: Nodes & edges are called **actors** & **channels**
- Implicit repetition, common in streaming, signal processing applications
- Communication: **only** through channels
- Multiple flavors of models: Rules that determine when an actor ***fires***
- A graph models multiple possible executions:

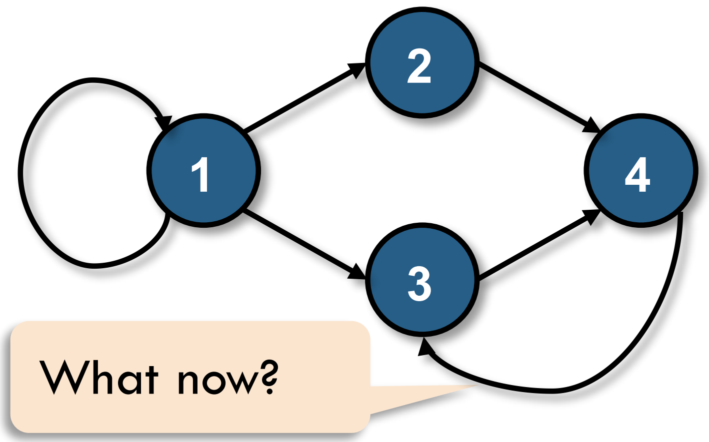


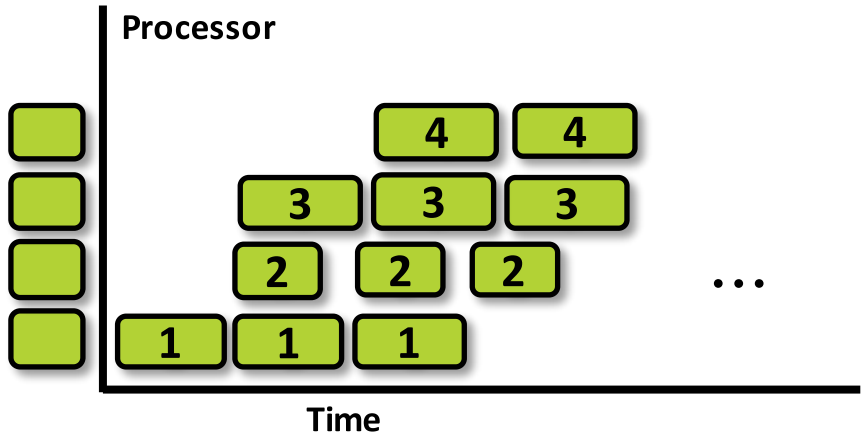


cf. LabVIEW models

Dataflow models (2)

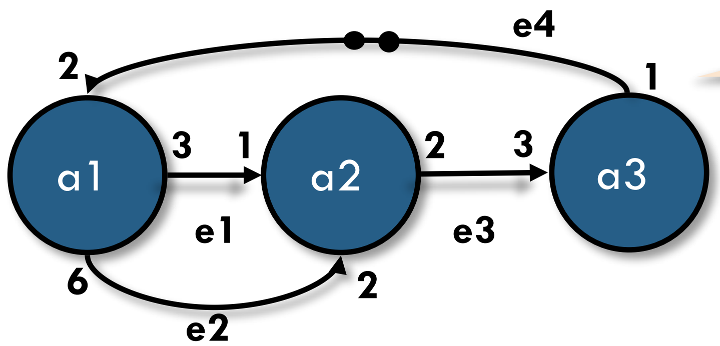
- Also a graph representation: Nodes & edges are called **actors** & **channels**
- Implicit repetition, common in streaming, signal processing applications
- Communication: **only** through channels
- Multiple flavors of models: Rules that determine when an actor ***fires***
- A graph models multiple possible executions:





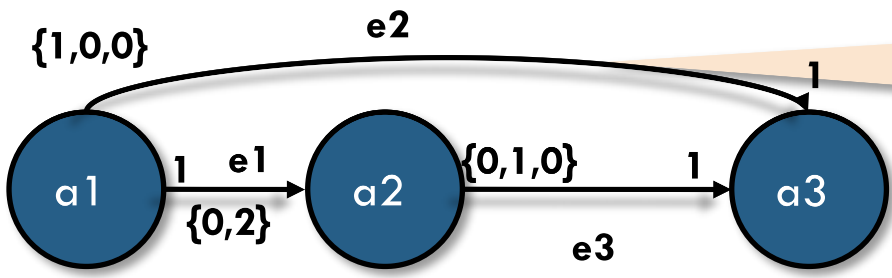
Dataflow models (3)

- Synchronous Dataflow (SDF): every actor has a **fixed behavior**



a3 always writes 1 token to e4

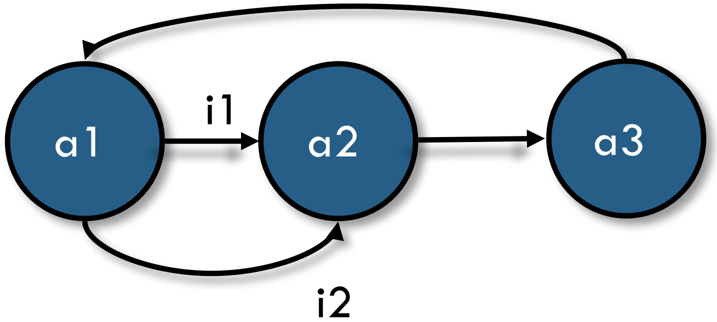
- Cyclo-Static Dataflow (CSDF): every actor has a **set of fixed behaviors**



a1: writes 1 token, then 0, then 0 to e2

Dynamic models and Kahn Process Networks

- Dynamic dataflow: set of firing rules per actor



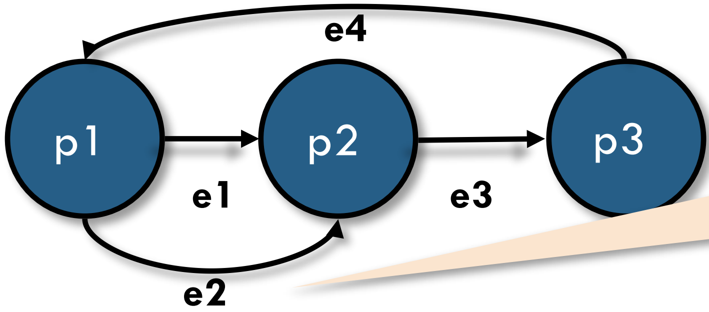
Firing rules:

...

$$R_{a_2,1} = ([*,*],[*])$$
$$R_{a_2,2} = (\perp,[0])$$
$$R_{a_2,3} = ([0],\perp)$$
$$R_{a_3,2} = ([*,*,*])$$

...

- Kahn process networks (KPN): nodes are now called processes



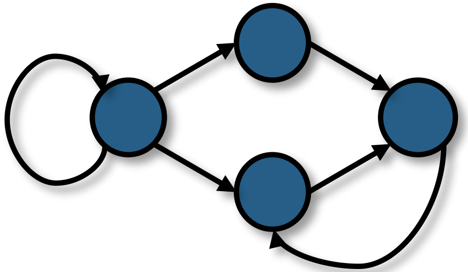
p1: writes any amount of tokens to e2 at any time

Outline

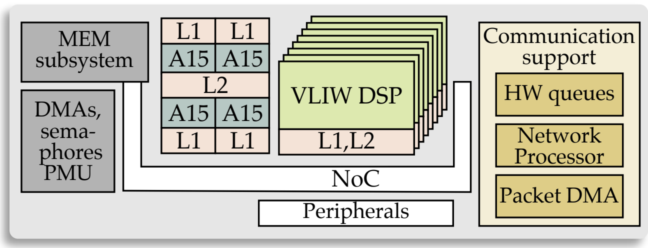
- Heterogeneous systems
- Dataflow models
- **Analysis and synthesis**
- Summary

Analysis and synthesis

Application



Architecture model



Non-functional specification

cf. Silexica's tool flows

Analysis

Synthesis

Code generation

Property models (timing, energy, error, …)

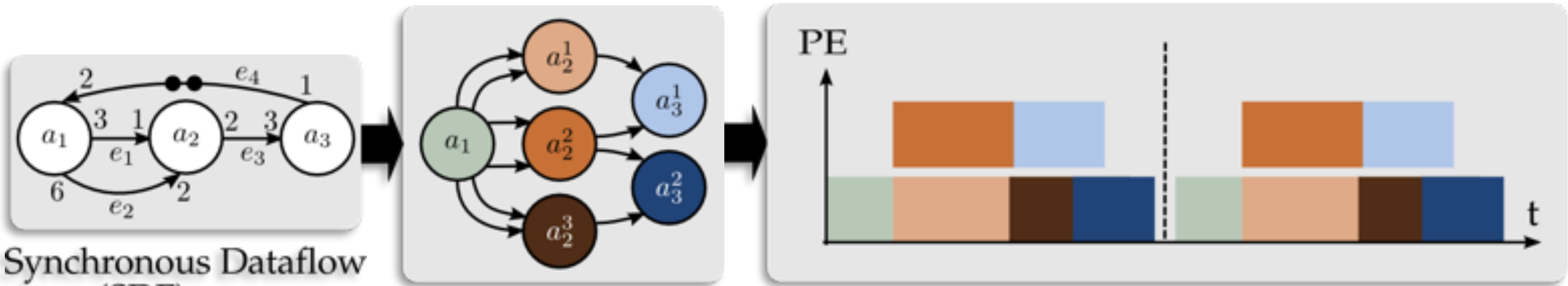
```
PNargs_ifft_r.ID = 6U;
   PNargs_ifft_r.PNchannel_freq_coef = f
   PNargs_ifft_r.PNnum_freq_coef = 0U;
   PNargs_ifft_r.PNchannel_time_coef = s
   PNargs_ifft_r.channel = 1;
   sink_left = IPCllmrf_open(3, 1, 1);
   sink_right = IPCllmrf_open(7, 1, 1);
   PNargs_sink.ID = 7U;
   PNargs_sink.PNchannel_in_left = sink_
   PNargs_sink.PNnum_in_left = 0U;
   PNargs_sink.PNchannel_in_right = sink
   PNargs_sink.PNnum_in_right = 0U;
   taskParams.arg0 = (xdc_UArg)&PNargs_s
   taskParams.priority = 1;

ti_sysbios_knl_Task_create((ti_sysbios_kr
&taskParams, &eb);
   glob_proc_cnt++;
   hasProcess = 1;
   taskParams.arg0 = (xdc_UArg)&PNargs_f
   taskParams.priority = 1;

ti_sysbios_knl_Task_create((ti_sysbios_kr
ft_Templ, &taskParams, &eb);
   glob_proc_cnt++;
   hasProcess = 1;
   taskParams.arg0 = (xdc_UArg)&PNargs_i
   taskParams.priority = 1;

ti_sysbios_knl_Task_create((ti_sysbios_kr
fft_Templ, &taskParams, &eb);
   glob_proc_cnt++;
   hasProcess = 1;
   taskParams.arg0 = (xdc_UArg)&PNargs_s
   taskParams.priority = 1;
```

Example for SDFs



Synchronous Dataflow (SDF) spec.

Unfolded cycle

"Blocked" DAG scheduling

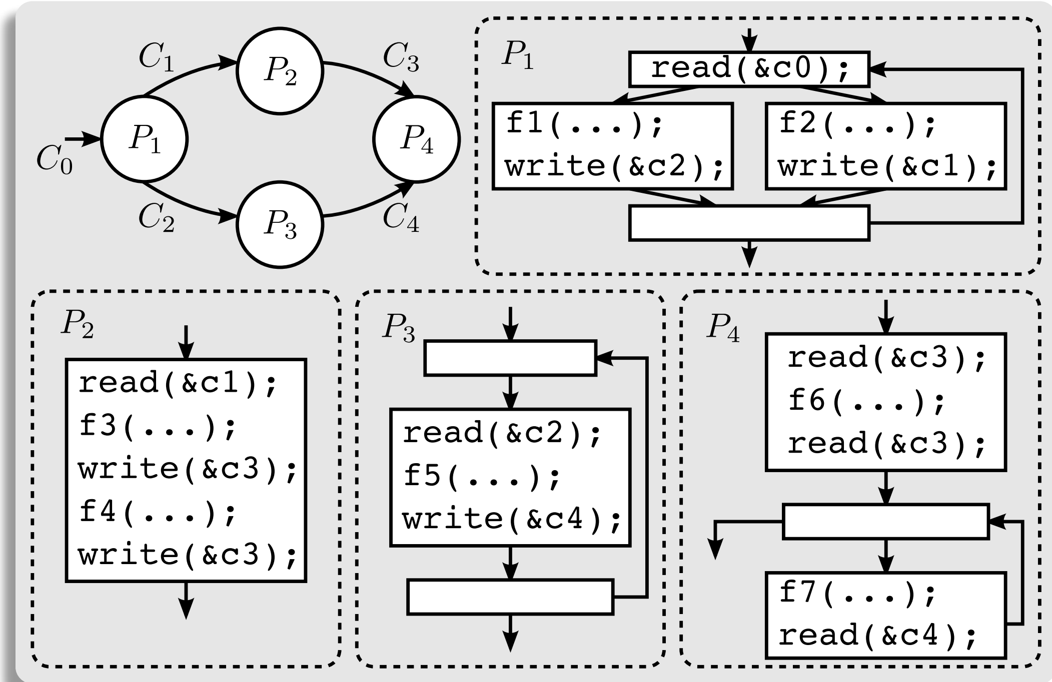
- Compute **topology matrix,** and solve system of equations
- Solution: **repetition vector** serve to unroll the graph [1 3 2]
- Perform mapping and scheduling on the resulting **directed acyclic graph** (DAG)

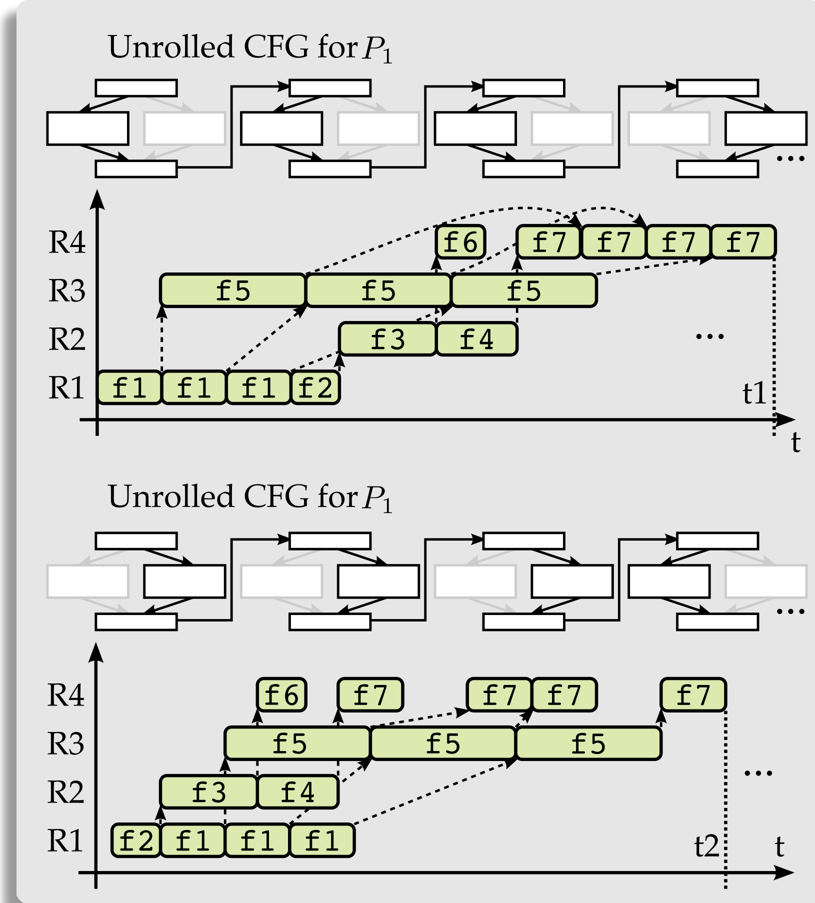
$$\Gamma = \begin{pmatrix} 3 & -1 & 0 \\ 6 & -2 & 0 \\ 0 & 2 & -3 \\ -2 & 0 & 1 \end{pmatrix}$$

$$\Gamma \cdot \mathbf{r} = 0$$

Example for KPNs: Static & dynamic analysis

- Need to understand process interactions

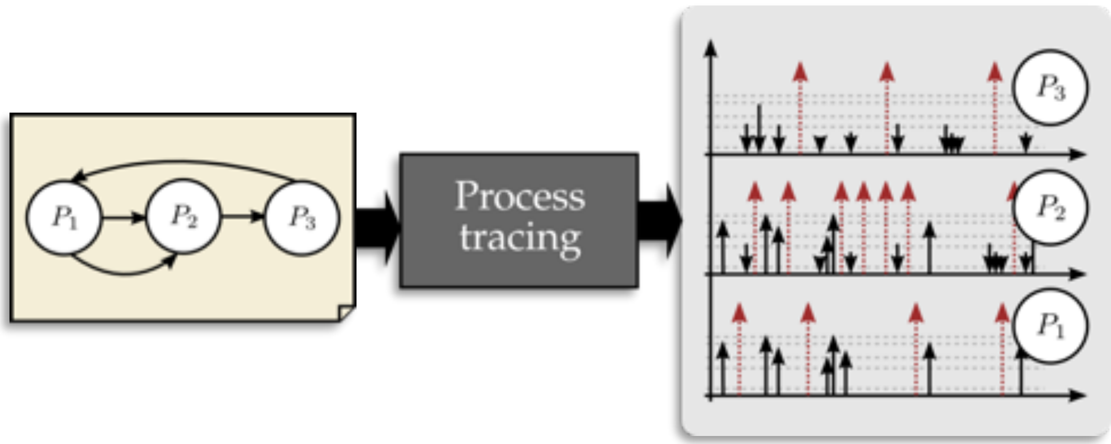




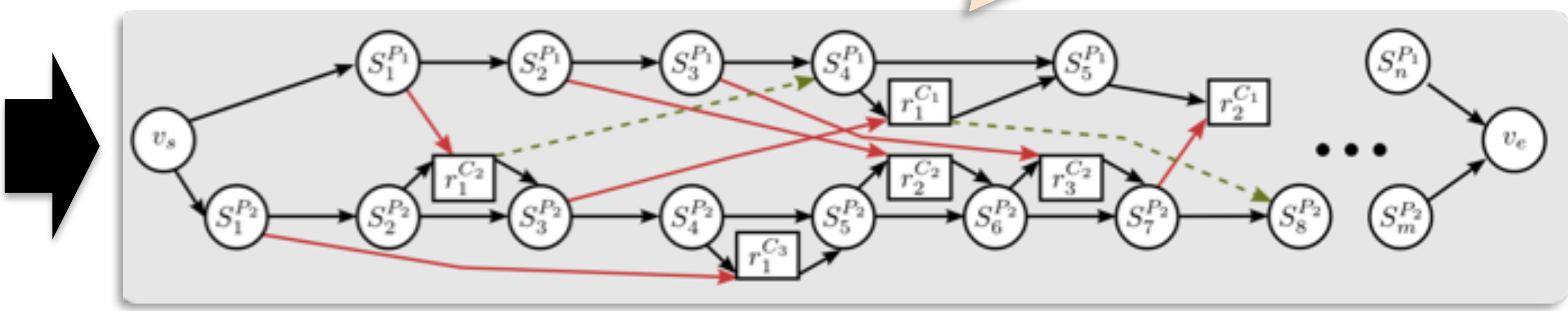
[Castrillon14]

KPN & DDF: Tracing

- Dynamic analysis based on execution traces [Castrillon10/13, Brunet15, Singh15]

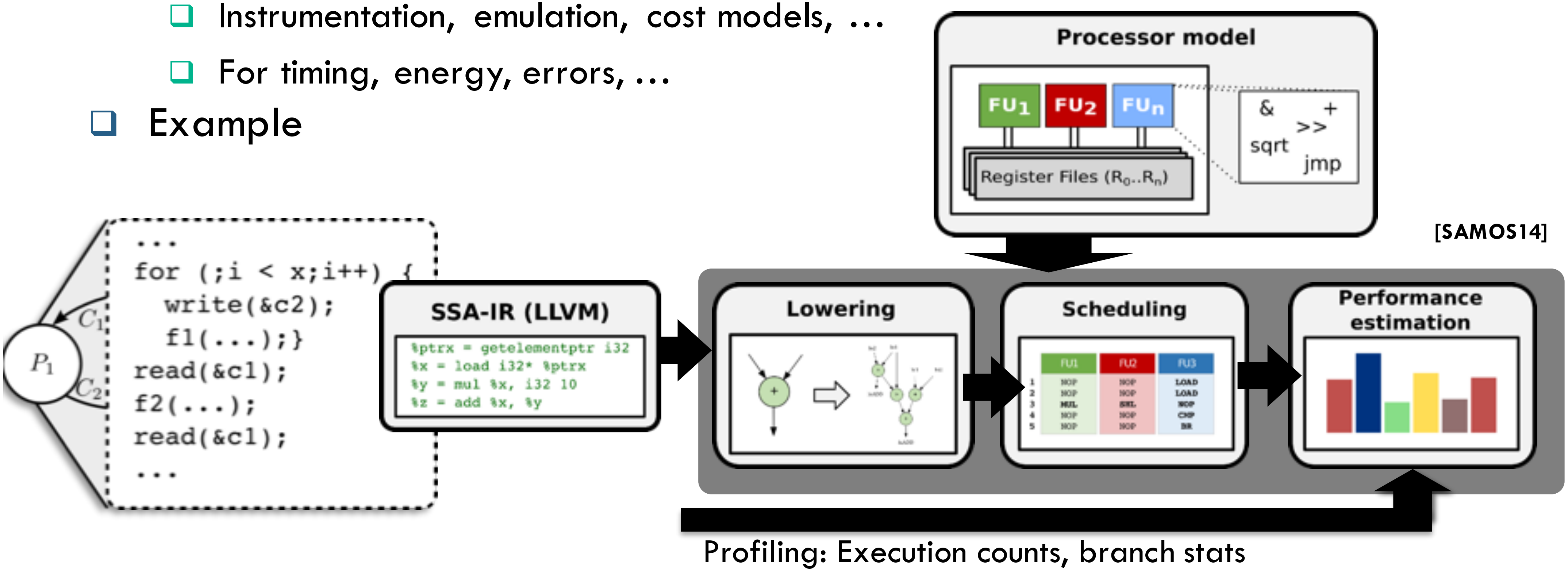


DAG representation for further analysis and synthesis



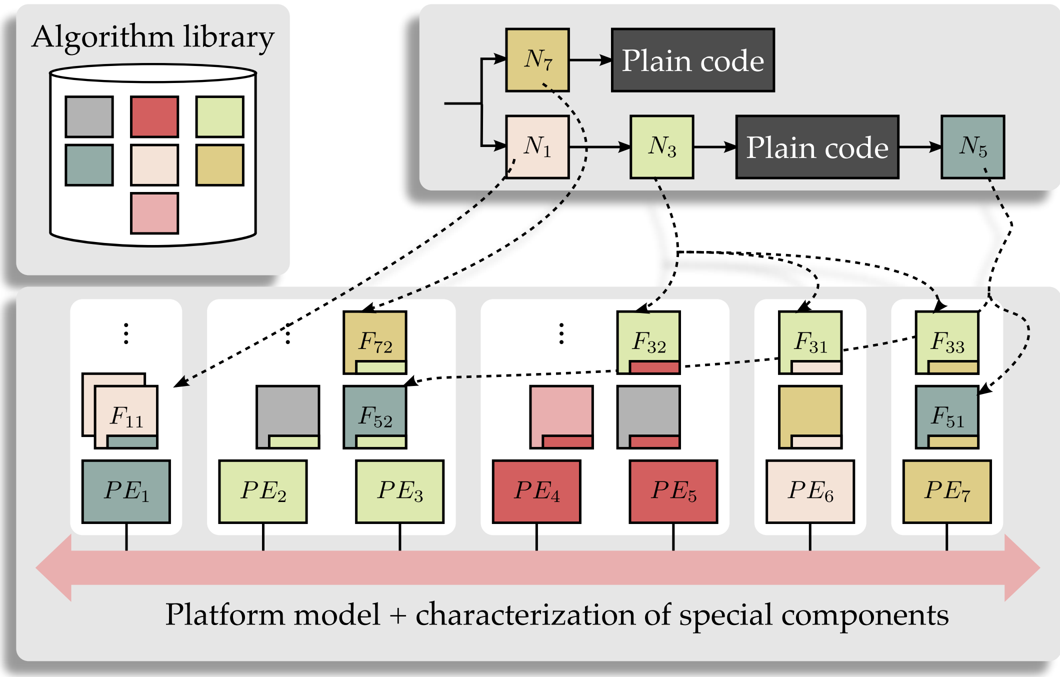
Models from functional specification

- Inspect functional specification of actors/processes (**cf. 2nd talk**)
  - Instrumentation, emulation, cost models, …
  - For timing, energy, errors, …
- Example



Models based on algorithmic descriptions

- When functional specification is not meant for synthesis
  - Common/required in heterogeneous systems (*special* components)
- Need to match algorithms to hardware components [Castrillon10b, Castrillon11]

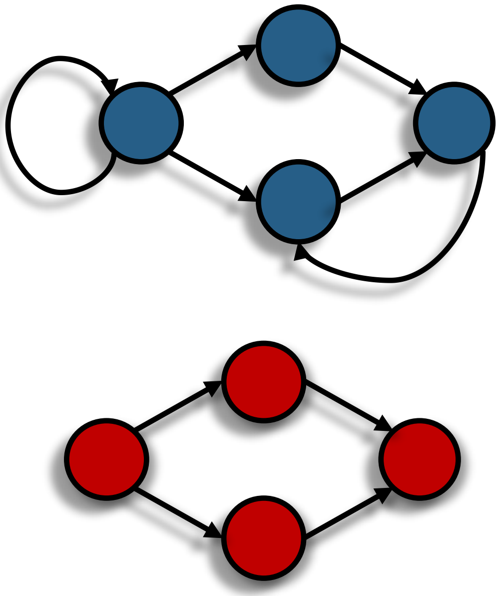


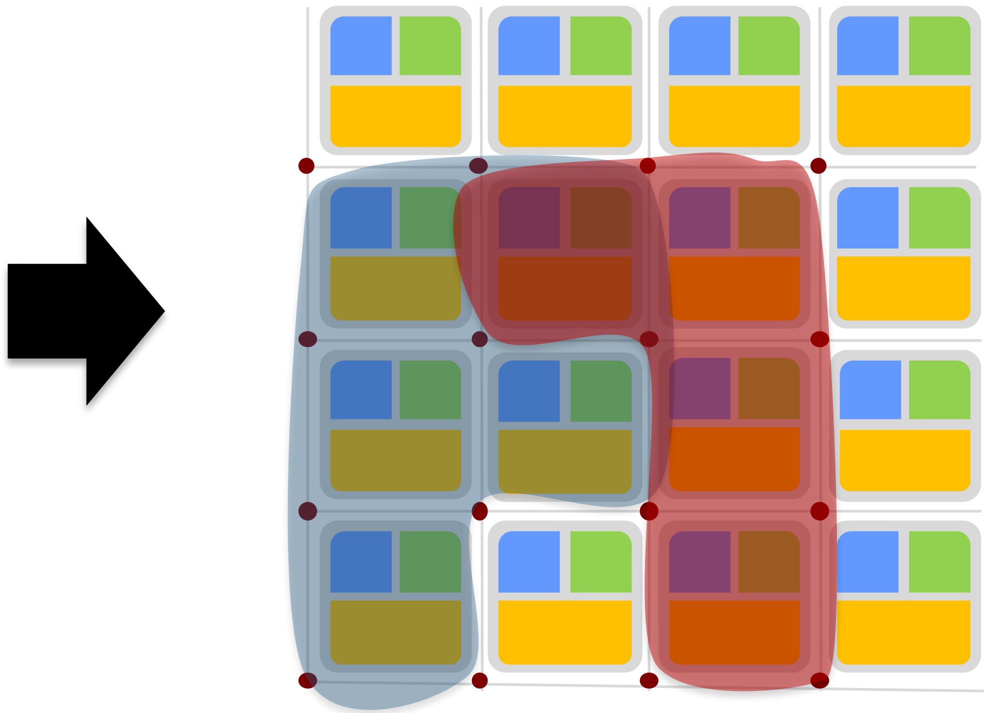
N: Algorithmic actors

F: Existing implementation in target platform

Multiple-applications

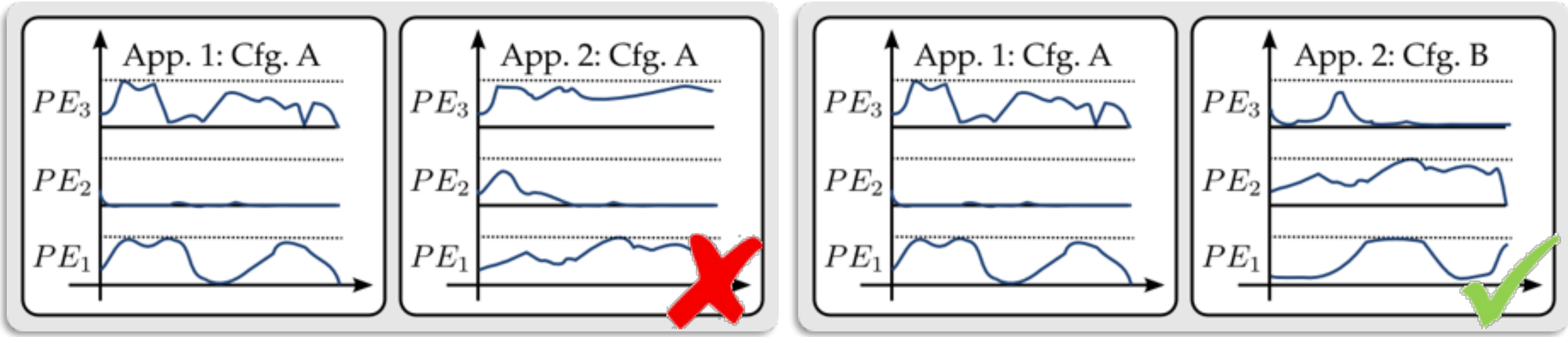
- Use traces and mappings to reason about platform sharing





Multiple-applications (2)

- Quickly discard "bad" multi-application configurations by observing the platform utilization profiles



Outline

- Heterogeneous systems
- Dataflow models
- Analysis and synthesis
- **Summary**

Summary

- Need programming models (and HW/SW stacks) to handle heterogeneity
- Even more dramatic in the post-CMOS era

- Dataflow models
  - Natural way to describe some applications
  - Amenable to analysis and synthesis for parallel execution
  - Discuss different kinds of models and required analysis
  - Need models of hardware for synthesis

Acknowledgements

- German Cluster of Excellence: Center for Advancing Electronics Dresden (www.cfaed.tu-dresden.de)
- Collaborative research center (CRC): Highly Adaptive Energy-Efficient Computing (HAEC)

CRC 912: Highly Adaptive Energy-Efficient Computing

- Silexica Software Solutions GmbH

References

[Trommer15] Trommer, et al. "Functionality-Enhanced Logic Gate Design Enabled by Symmetrical Reconfigurable Silicon Nanowire Transistors", IEEE Trans on in Nanotechnology, pp.689-698, July 2015

[Voigt14] Andreas Voigt, et al. "Towards Computation with Microchemomechanical Systems", In International Journal of Foundations of Computer Science, World Scientific, volume 25, 2014

[Augonnet10] C. Augonnet, et al. "StarPU: a unified platform for task scheduling on heterogeneous multicore architectures", Concurrency and Computation: Practice and Experience 23.2 (2011), pp. 187–198.

[Kwok99] Y.-K. Kwok, et al, "Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors", ACM Comput. Surv., ACM Press, 1999, 31, 406 - 471

[Castrillon14] J. Castrillon, et al, "Programming Heterogeneous MPSoCs: Tool Flows to Close the Software Productivity Gap", Springer, 2014

[Castrillon10] J. Castrillon, et al, "Trace-based KPN composability analysis for mapping simultaneous applications to MPSoC platforms", DATE'10, pp. 753-758, 2010

[Castrillon13] J. Castrillon, et al, "MAPS: Mapping concurrent dataflow applications to heterogeneous MPSoCs," IEEE Trans on Industrial Informatics, vol. 9, no. 1, pp. 527–545, 2013

[Brunet15] Brunet, Simone C. "Analysis and optimization of dynamic dataflow programs". Diss. ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2015

[Singh15] Singh, Amit, et al. "Resource and Throughput Aware Execution Trace Analysis for Efficient Run-time Mapping on MPSoCs.", TCAD 2015

[SAMOS14] J.F. Eusse, et al, "Pre-architectural performance estimation for ASIP design based on abstract processor models," *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), 2014* pp.133-140, 2014

[Castrillon10b] J. Castrillon, et al, "Component-based waveform development: The nucleus tool flow for efficient and portable SDR," Wireless Innovation Conference and Product Exposition (SDR), 2010

[Castrillon11] J. Castrillon, et al, "Component-based waveform development: The nucleus tool flow for efficient and portable software defined radio", Analog Integrated Circuits and Signal Processing, vol. 69, no. 2–3, pp. 173–190, 2011