

Portable Libraries and Programming Environments

Jeronimo Castrillon

Chair for Compiler Construction

TU Dresden

jeronimo.castrillon@tu-dresden.de

Thematic session: Towards Portable Libraries for Hybrid Systems

HiPEAC Computing Systems Week. Oslo, May 6th 2015

Acknowledgements



- ❑ German Cluster of Excellence: Center for Advancing Electronics Dresden (www.cfaed.tu-dresden.de)



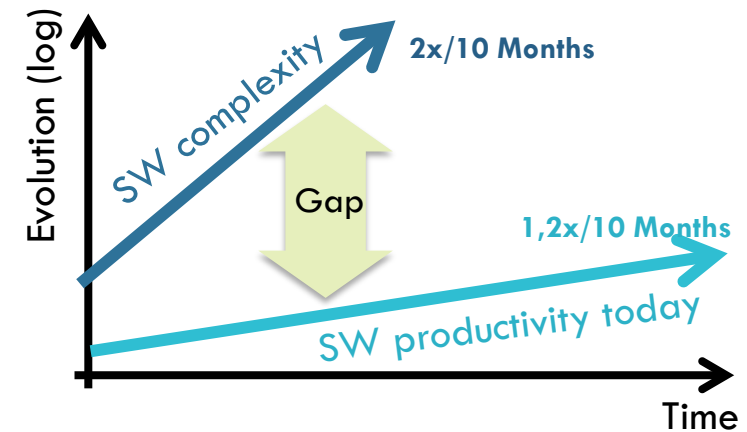
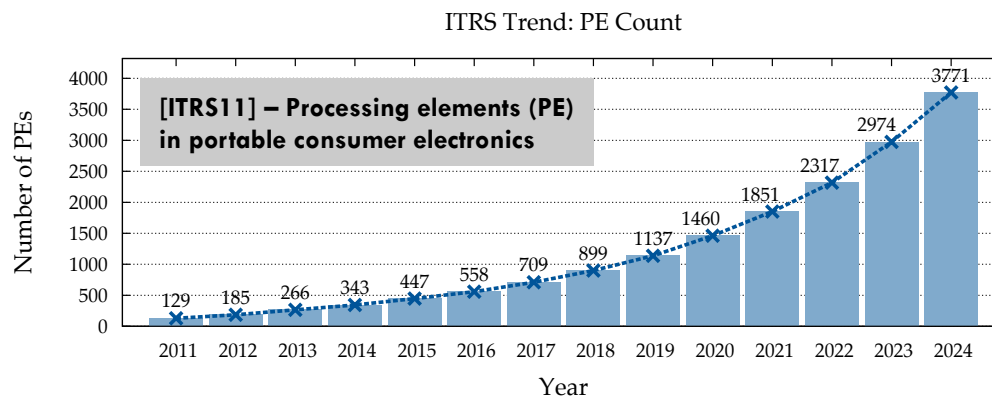
- ❑ Silexica Software Solutions GmbH



- ❑ German Cluster of Excellence: Ultra High-Speed Mobile Information and Communication (UMIC), RWTH Aachen



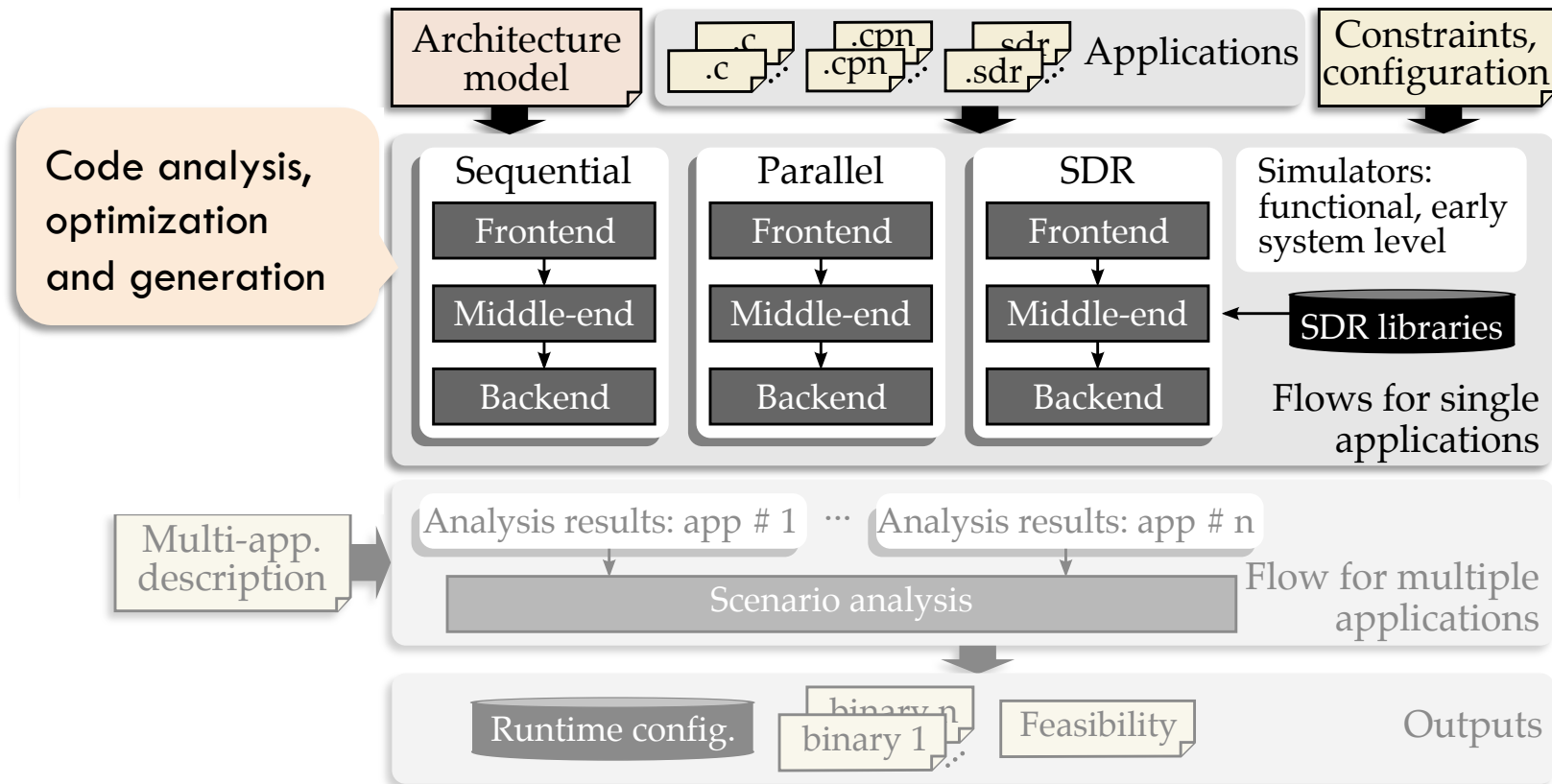
Why new programming environments?



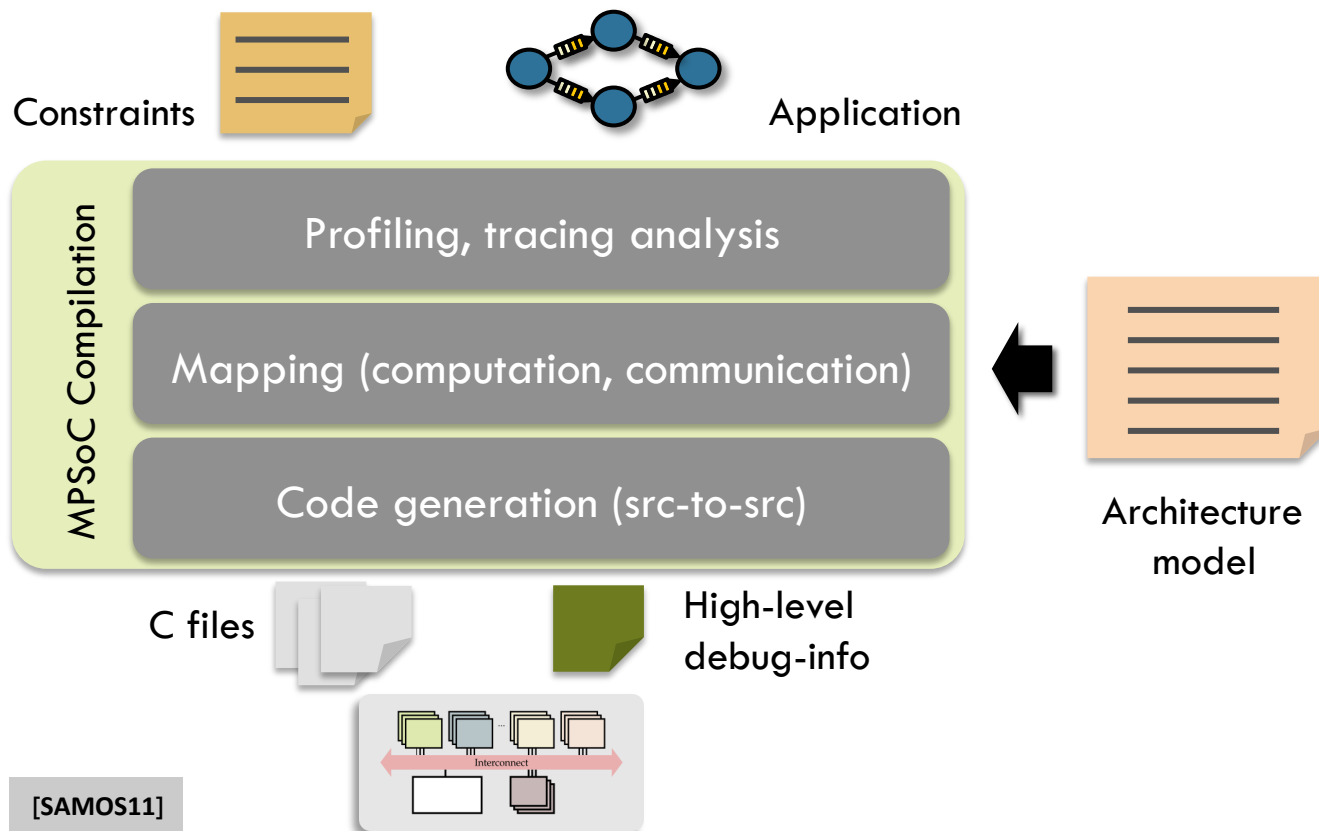
- ❑ Complex applications for complex architectures → SW-productivity gap
- ❑ Already difficult for heterogeneous multi-core (programmable) systems
- ❑ More difficult for systems with hardware accelerators and reconfigurable HW

Programming flows for heterogeneous MPSoCs

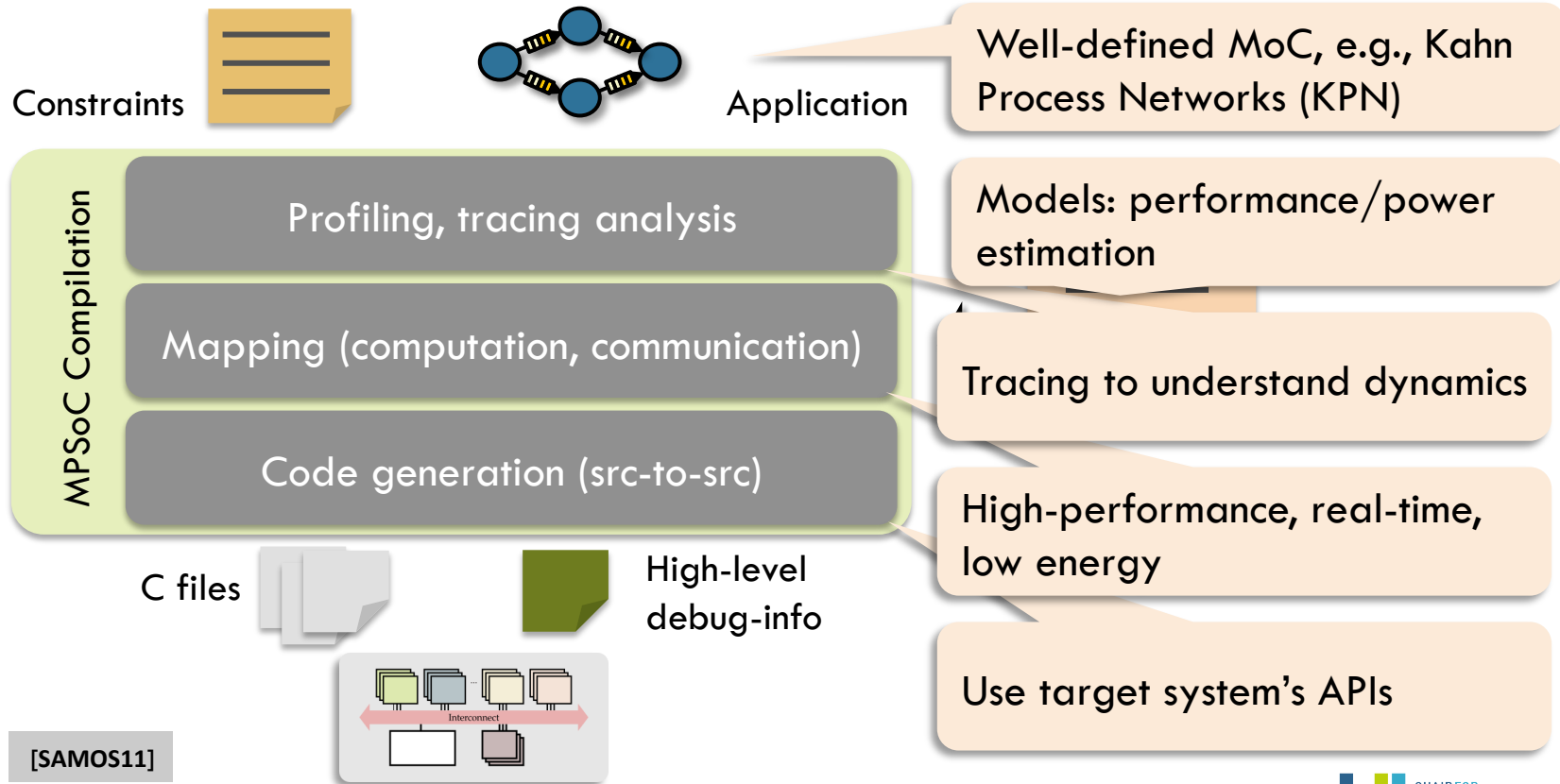
[Castrillon14]



Parallel programming flow: Overview

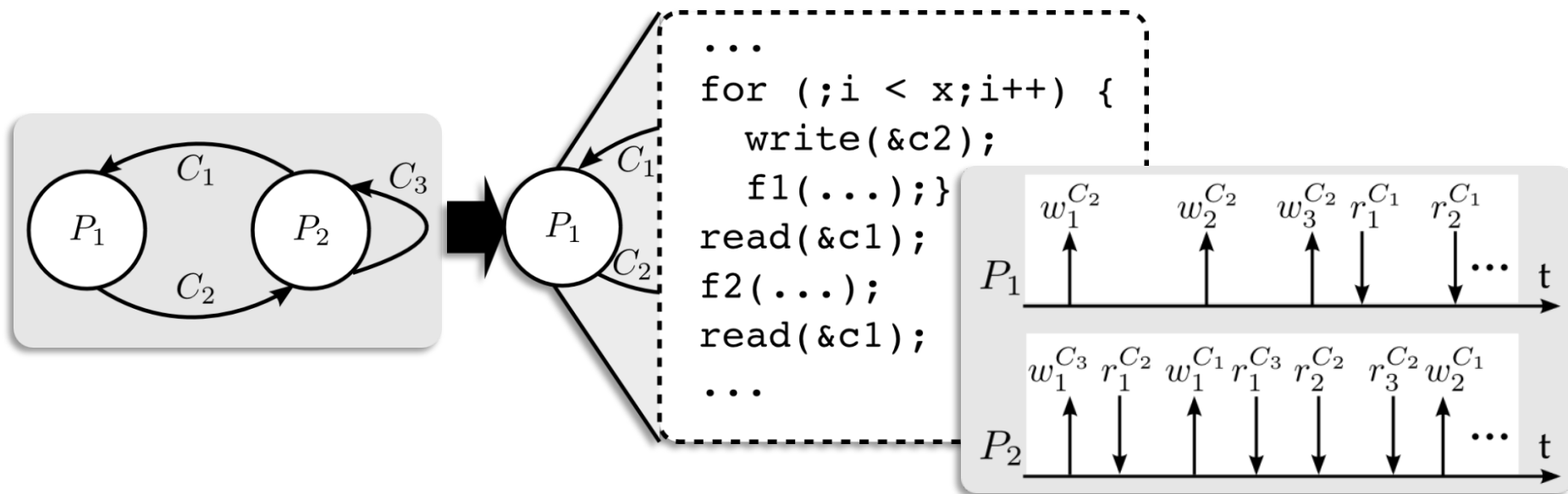


Parallel programming flow: Overview

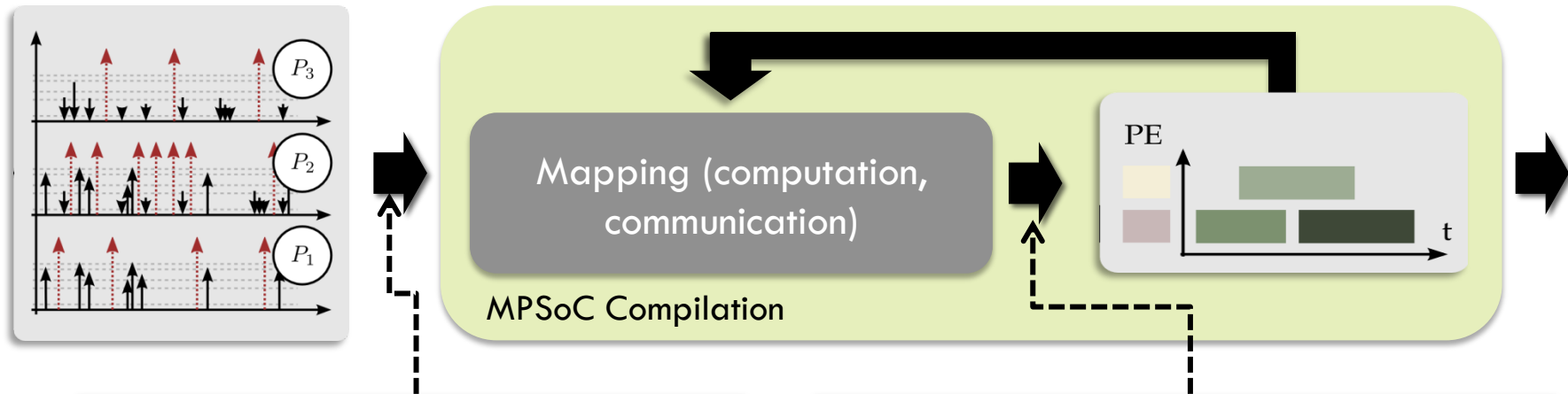


Analysis: Instrumentation, profiling and tracing

- **Process white model:** source code analysis and tracing to deal with control



Performance models: Sequential and parallel



Sequential performance estimation

- Design ↓
- Annotations & cost functions
 - Abstract operation cost models
 - Processor models/simulators
 - Measurements

Parallel performance estimation

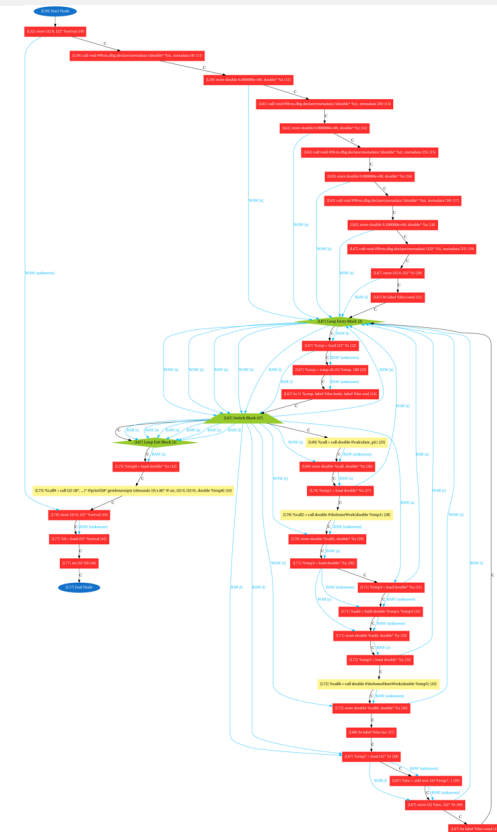
- Abstract cost models: OS, multi-tasking APIs, interconnect & memories
- System simulators/emulators
- Boards

Performance estimation: Computation

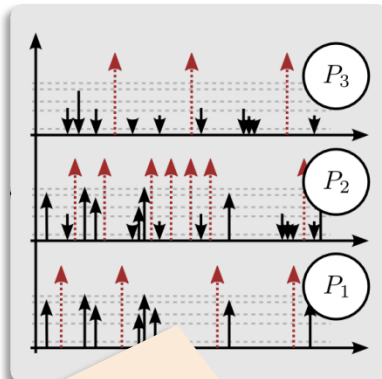
- ❑ Cost tables
 - ❑ Equivalence: Low-level IR \rightarrow Assembly instructions
 - ❑ Coarse estimation of instruction-level parallelism
- ❑ Processor + compiler models
 - ❑ Set of resources, operations, low-level APIs

FU₁	FU₂	FU₃	Conventions Prologue: $1+2 \cdot X_{arg}$ Epilogue: 4 Branch _{overh} : 3
ADD SUB MUL OR	ASR LSL LSR ZEX	LD ST CMP BR	
↕	↕	↕	External library costs malloc: $9+0.3 \cdot N_{size}$ fsqrt: 235
RF₁(16,32)	RF₂(8,32)		

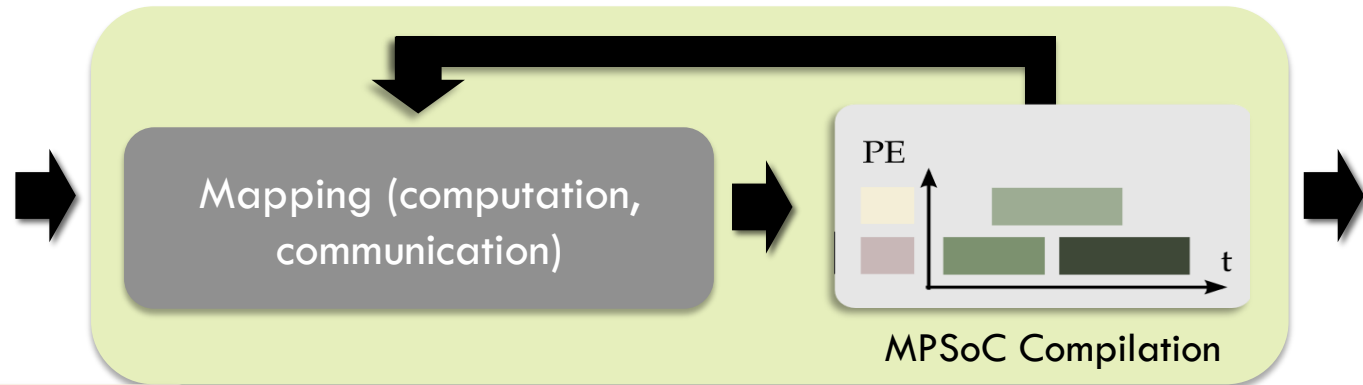
[SAMOS14]
Courtesy: J. Eusse



Application mapping: Trace-based



Traces + timing information
from platform models

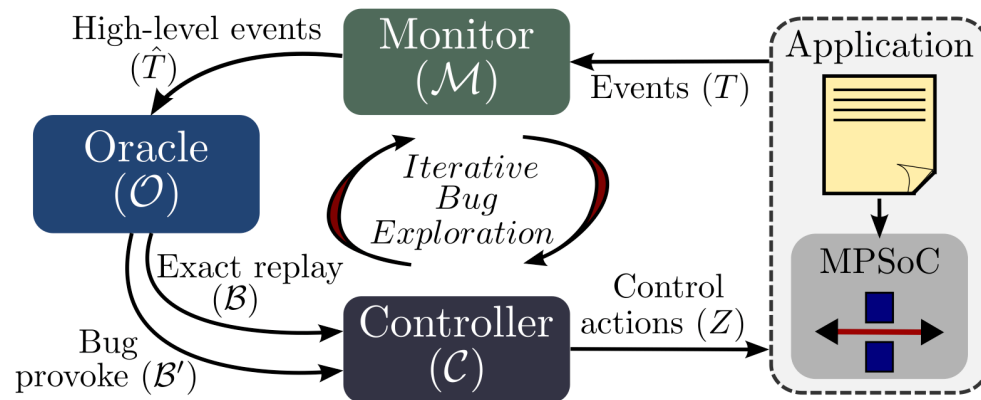


- ❑ Heuristics that work on a graph-representation of multiple-traces
 - ❑ For buffer sizing
 - ❑ For mapping and scheduling
 - ❑ Often iterative for real-time

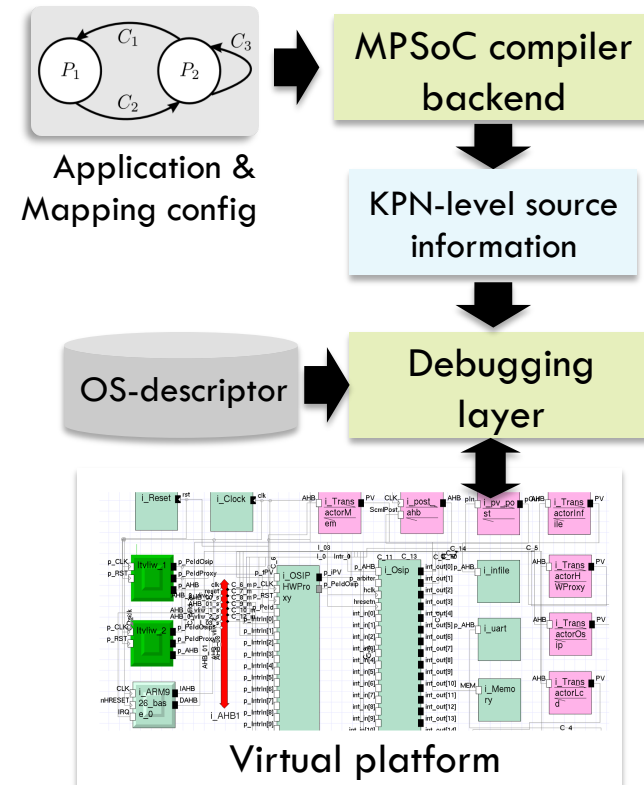
[DATE10, DAC12, IEEE-TII13]

Debugging: Layers and scripting for virtual platforms

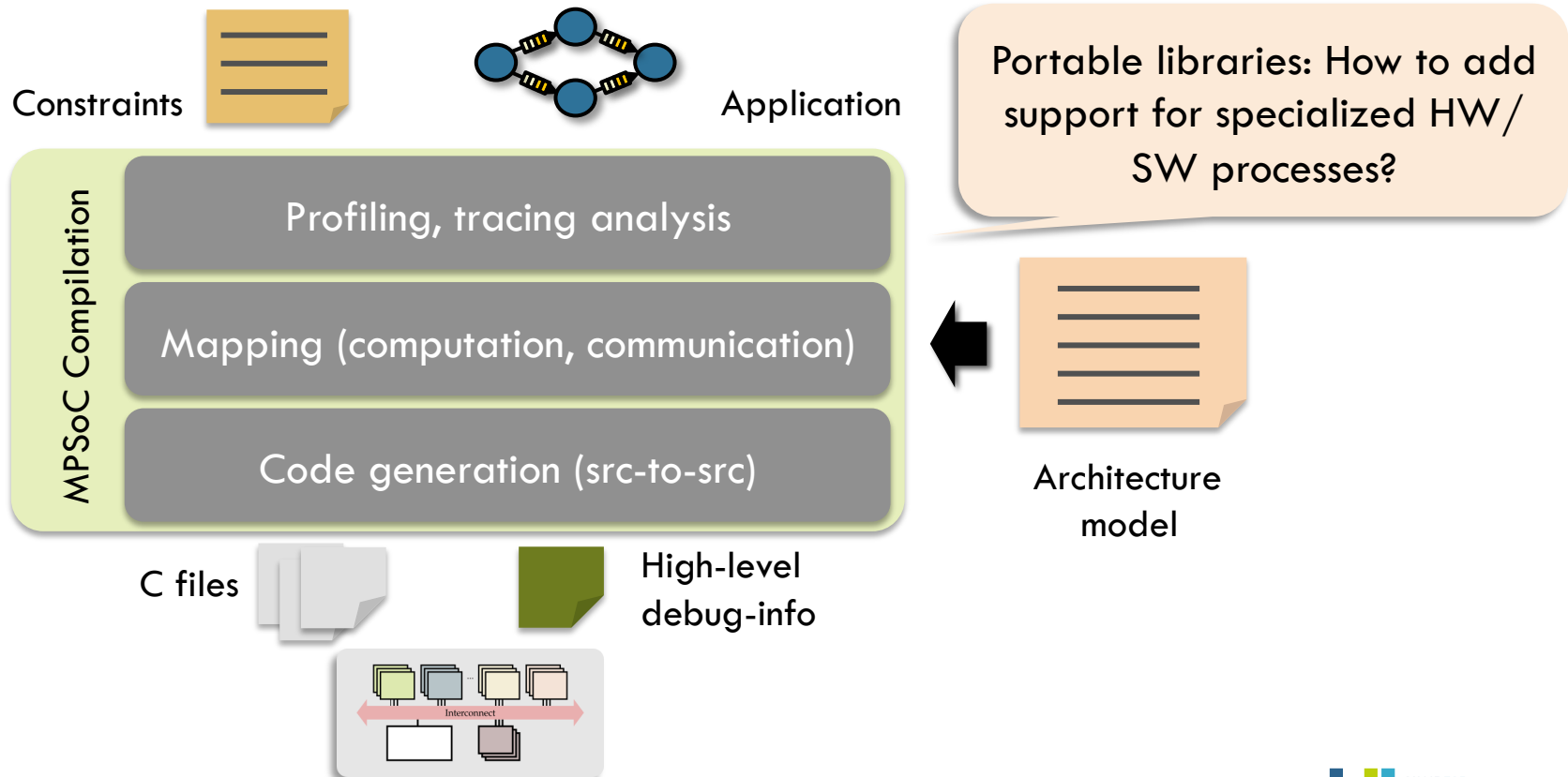
- ❑ Interactive debugging
- ❑ Deterministic replay for bug exploration



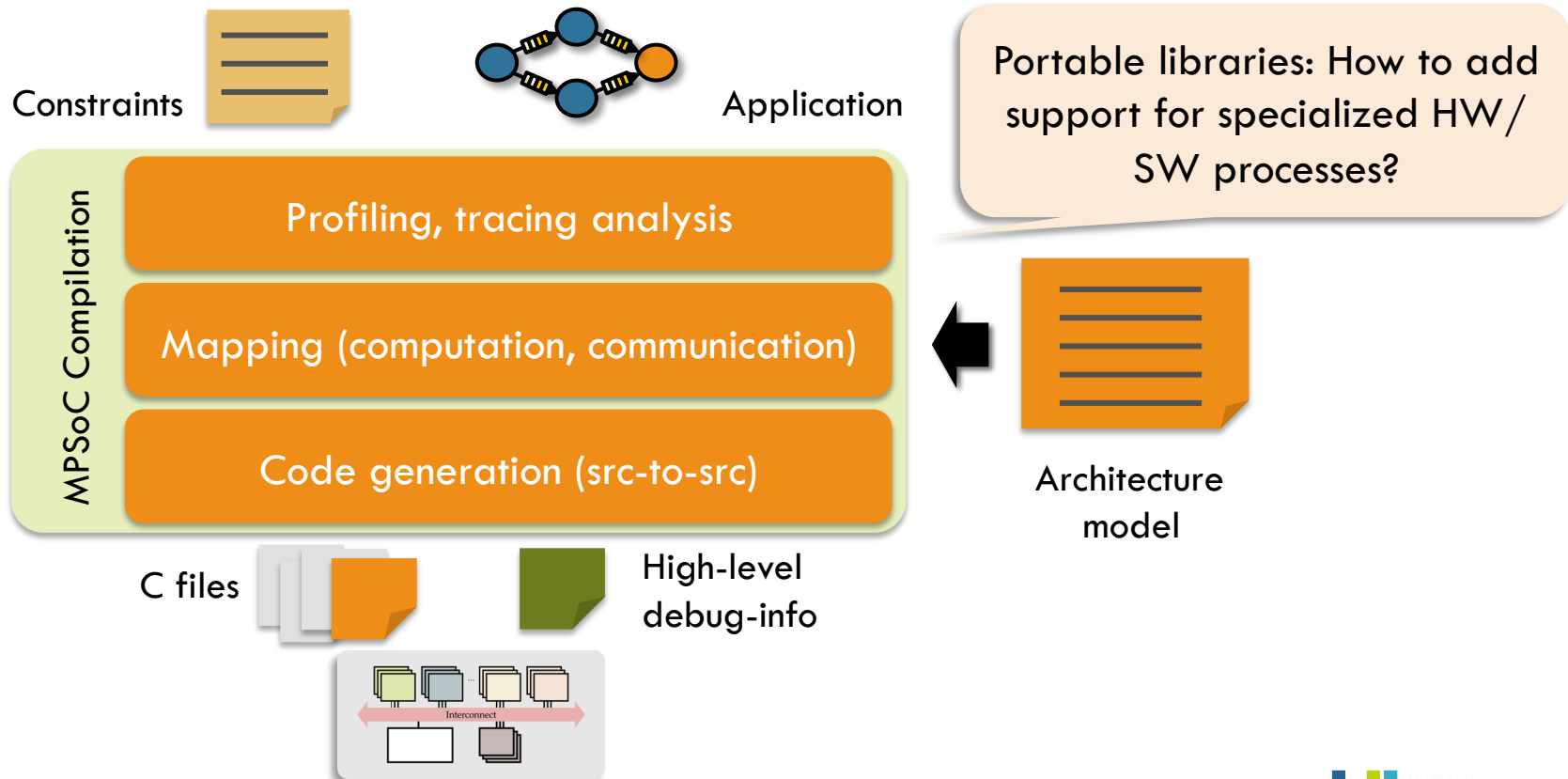
[LASCAS10, DATE14]



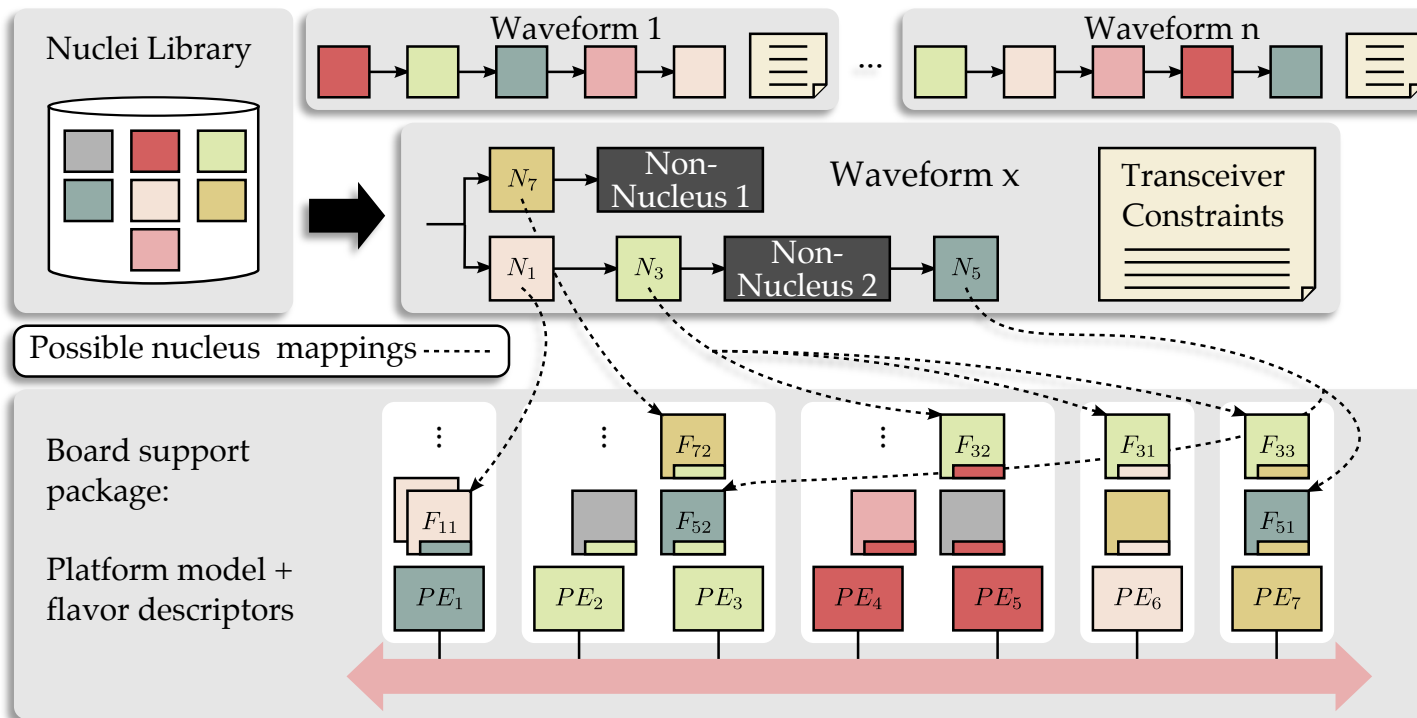
Parallel programming flow: Portable libraries



Parallel programming flow: Portable libraries



Solution approach: Nucleus project (for SW-defined Radio)

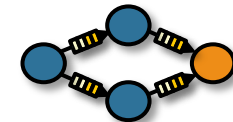


[SDR10, ALOG11, Castrillon14]

Nucleus programming flow: Inputs

- ❑ Extended application specification
 - ❑ Selected processes are algorithmic kernels with **algorithmic parameters**

- ❑ Extended platform model
 - ❑ SW/HW accelerated kernels and their **implementation parameters**



- Types & parameters
- Latency equations
- Interfacing

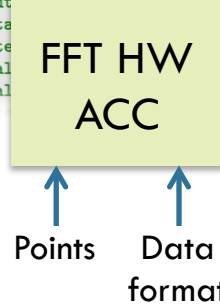
Nucleus programming flow: Mapping

- Extended mapping

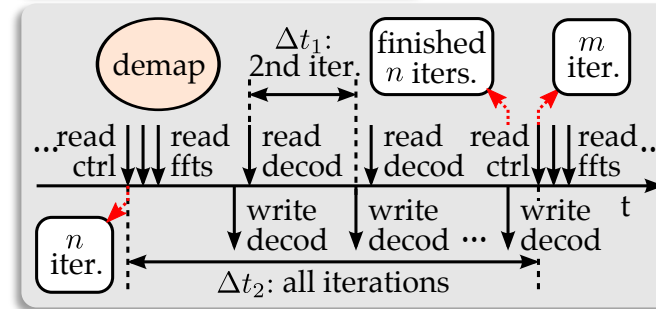
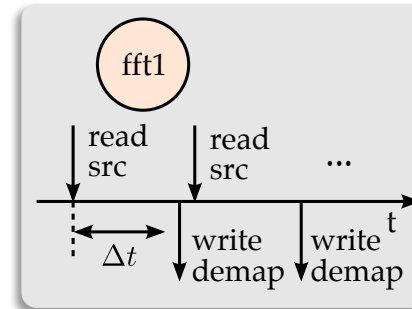
- Find matching HW/SW support: Algorithmic to implementation parameters
- Integrate into trace-based framework

```

1 <Flavor name="fft_HW" Nucleus="FFT">
2   <parameter name="bitwidth"><value>32.0</value></parameter>
3   <parameter name="points"><values List="16,32,64,128,256,512,1024,2048,4096,8192,16384"></values List></parameter>
4   <property name="latency"><function>16 * points</function></property>
5   <input name="src" input</port>
6   <Data type="fixed_point" data="" data_out="" data_in="" data_out2="" data_in2="" data_in3="" data_out3="" data_in4="" data_out4="" data_in5="" data_out5="" data_in6="" data_out6="" data_in7="" data_out7="" data_in8="" data_out8="" data_in9="" data_out9="" data_in10="" data_out10="" data_in11="" data_out11="" data_in12="" data_out12="" data_in13="" data_out13="" data_in14="" data_out14="" data_in15="" data_out15="" data_in16="" data_out16="" data_in17="" data_out17="" data_in18="" data_out18="" data_in19="" data_out19="" data_in20="" data_out20="" data_in21="" data_out21="" data_in22="" data_out22="" data_in23="" data_out23="" data_in24="" data_out24="" data_in25="" data_out25="" data_in26="" data_out26="" data_in27="" data_out27="" data_in28="" data_out28="" data_in29="" data_out29="" data_in30="" data_out30="" data_in31="" data_out31="" data_in32="" data_out32="" data_in33="" data_out33="" data_in34="" data_out34="" data_in35="" data_out35="" data_in36="" data_out36="" data_in37="" data_out37="" data_in38="" data_out38="" data_in39="" data_out39="" data_in40="" data_out40="" data_in41="" data_out41="" data_in42="" data_out42="" data_in43="" data_out43="" data_in44="" data_out44="" data_in45="" data_out45="" data_in46="" data_out46="" data_in47="" data_out47="" data_in48="" data_out48="" data_in49="" data_out49="" data_in50="" data_out50="" data_in51="" data_out51="" data_in52="" data_out52="" data_in53="" data_out53="" data_in54="" data_out54="" data_in55="" data_out55="" data_in56="" data_out56="" data_in57="" data_out57="" data_in58="" data_out58="" data_in59="" data_out59="" data_in60="" data_out60="" data_in61="" data_out61="" data_in62="" data_out62="" data_in63="" data_out63="" data_in64="" data_out64="" data_in65="" data_out65="" data_in66="" data_out66="" data_in67="" data_out67="" data_in68="" data_out68="" data_in69="" data_out69="" data_in70="" data_out70="" data_in71="" data_out71="" data_in72="" data_out72="" data_in73="" data_out73="" data_in74="" data_out74="" data_in75="" data_out75="" data_in76="" data_out76="" data_in77="" data_out77="" data_in78="" data_out78="" data_in79="" data_out79="" data_in80="" data_out80="" data_in81="" data_out81="" data_in82="" data_out82="" data_in83="" data_out83="" data_in84="" data_out84="" data_in85="" data_out85="" data_in86="" data_out86="" data_in87="" data_out87="" data_in88="" data_out88="" data_in89="" data_out89="" data_in90="" data_out90="" data_in91="" data_out91="" data_in92="" data_out92="" data_in93="" data_out93="" data_in94="" data_out94="" data_in95="" data_out95="" data_in96="" data_out96="" data_in97="" data_out97="" data_in98="" data_out98="" data_in99="" data_out99="" data_in100="" data_out100="" data_in101="" data_out101="" data_in102="" data_out102="" data_in103="" data_out103="" data_in104="" data_out104="" data_in105="" data_out105="" data_in106="" data_out106="" data_in107="" data_out107="" data_in108="" data_out108="" data_in109="" data_out109="" data_in110="" data_out110="" data_in111="" data_out111="" data_in112="" data_out112="" data_in113="" data_out113="" data_in114="" data_out114="" data_in115="" data_out115="" data_in116="" data_out116="" data_in117="" data_out117="" data_in118="" data_out118="" data_in119="" data_out119="" data_in120="" data_out120="" data_in121="" data_out121="" data_in122="" data_out122="" data_in123="" data_out123="" data_in124="" data_out124="" data_in125="" data_out125="" data_in126="" data_out126="" data_in127="" data_out127="" data_in128="" data_out128"
7   <Interface name="fft1" data="" data_out="" data_in="" data_out2="" data_in2="" data_in3="" data_out3="" data_in4="" data_out4="" data_in5="" data_out5="" data_in6="" data_out6="" data_in7="" data_out7"
8   <val name="ag_1of2" /></Interface>
9   <val name="ag_1of2" /></Interface>
10  </Flavor>
  
```

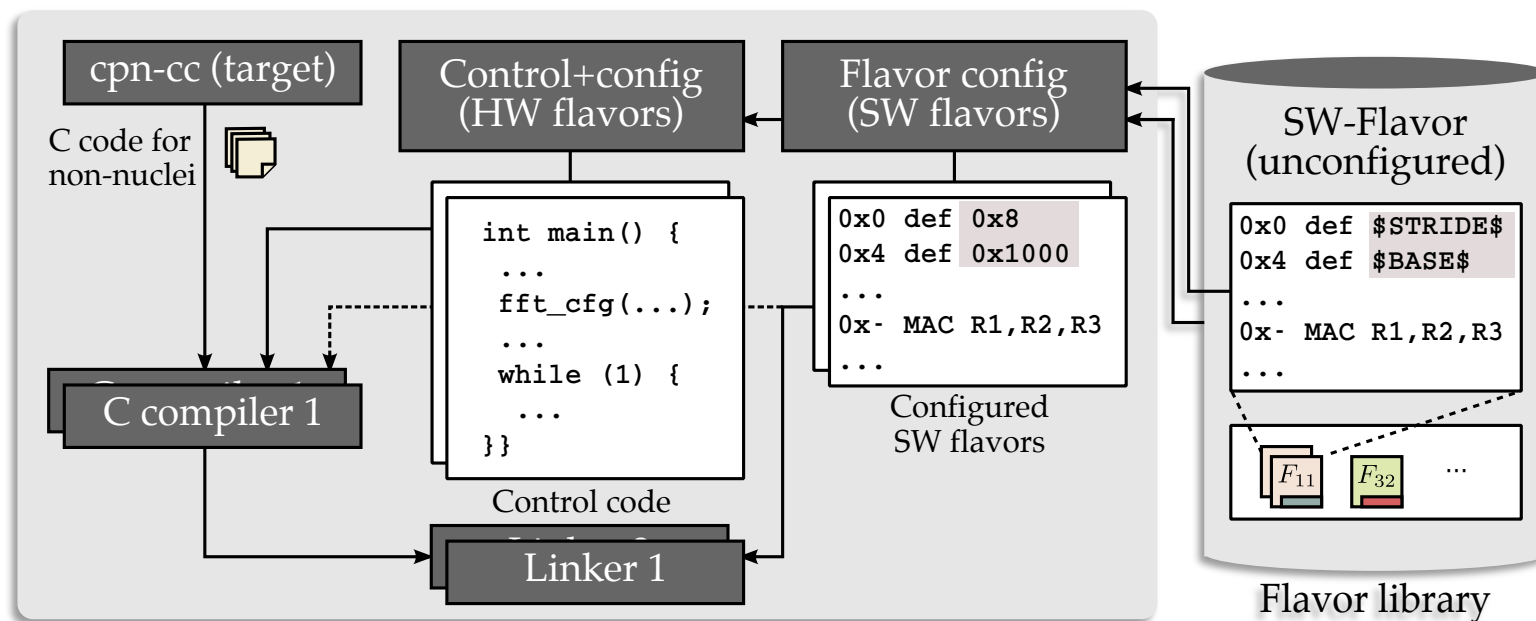


Configuration,
characterization



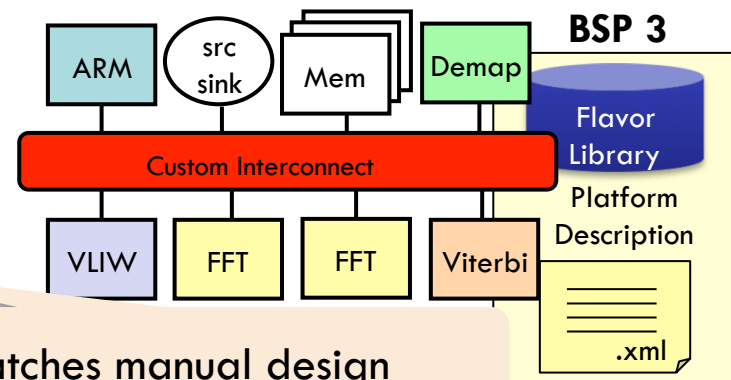
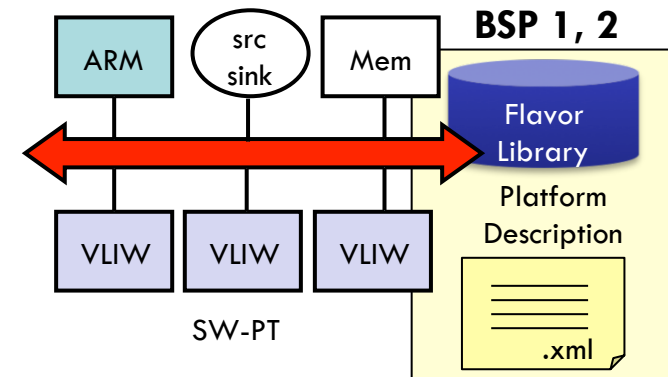
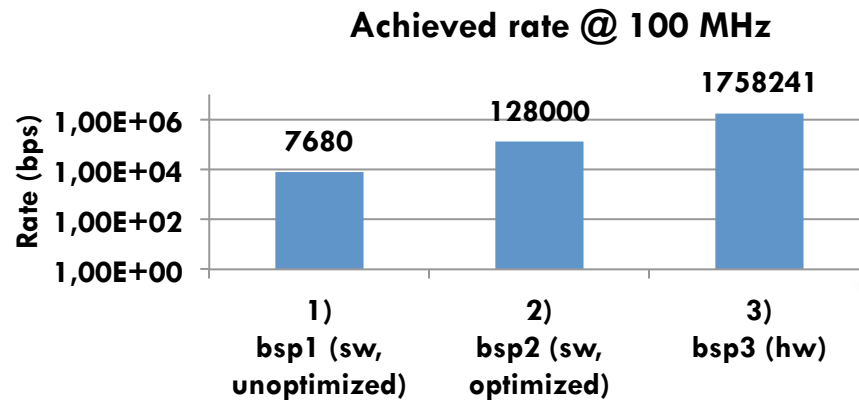
Nucleus programming flow: Code generation

□ Bare-metal implementation



Case study (brief): Portable performance

- ❑ Application: MIMO OFDM receiver
- ❑ Hardware
 - ❑ Platform 1: Baseline software
 - ❑ Platform 2: Optimized software
 - ❑ Platform 3: Optimized SW + HW



Matches manual design

Discussion towards library portability

- ❑ Programming flow extension to deal with HW/SW acceleration
- ❑ Desired library characterization (standards?)
 - ❑ For mapping
 - Times, data-types, rates (and more complex behaviors – time diagrams?)
 - ❑ For debugging
 - Debugging interfaces and true integration (so far: bare metal)
 - Interfacing patterns to be proven by trace analysis
 - ❑ For code generation (so far: bare metal)
 - OS integration, interaction with resource manager
- ❑ Not addressed: Reconfiguration & synthesis

References



- [ITRS11] International Technology Roadmap for Semiconductors (ITRS), “System Drivers,” 2011. [Online]. Available: <http://www.itrs.net/>
- [Castrillon14] J. Castrillon and R. Leupers, Programming Heterogeneous MPSoCs: Tool Flows to Close the Software Productivity Gap. Springer, 2014
- [SAMOS11] J. Castrillon, W. Sheng and R. Leupers, “Trends in embedded software synthesis” *Embedded Computer Systems (SAMOS), 2011 International Conference on*, 2011, 347-354
- [SAMOS14] J.F. Eusse, C. Williams, L.G. Murillo, R. Leupers and G. Ascheid, G., "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
- [DATE10] J. Castrillon, R. Velasquez, A. Stulova, W. Sheng, J. Ceng, R. Leupers, G. Ascheid, H. Meyr, “Trace-based KPN composability analysis for mapping simultaneous applications to MPSoC platforms”, *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 753-758, 2010
- [DAC12] J. Castrillon, A. Tretter, R. Leupers, and G. Ascheid, “Communication-aware mapping of KPN applications onto heterogeneous MPSoCs,” in *Proceedings of the Design Automation Conference*, 2012
- [IEEE-TII13] J. Castrillon, R. Leupers, and G. Ascheid, “MAPS: Mapping concurrent dataflow applications to heterogeneous MPSoCs,” *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 527–545, 2013
- [SOC13] M. Odendahl, J. Castrillon, V. Volevach, R. Leupers and G. Ascheid, “Split-cost communication model for improved MPSoC application mapping”, In *International Symposium on System on Chip (SoC)* pp. 1-8, 2013
- [SDR10] J. Castrillon, S. Schürmans, A. Stulova, W. Sheng, T. Kempf, R. Leupers, G. Ascheid, and H. Meyr, “Component-based waveform development: The nucleus tool flow for efficient and portable SDR,” *Wireless Innovation Conference and Product Exposition (SDR)*, 2010
- [ALOG11] J. Castrillon, S. Schürmans, A. Stulova, W. Sheng, T. Kempf, R. Leupers, G. Ascheid, and H. Meyr, “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
- [LASCAS14] J. Castrillon, A. Shah, L. G. Murillo, R. Leupers, G. Ascheid, “Backend for virtual platforms with hardware scheduler in the MAPS framework”, *Latin American Symposium on Circuits and Systems (LASCAS)*, pp. 1-4, 2011
- [DATE14] L. G. Murillo, S. Wawroschek, J. Castrillon, R. Leupers, G. Ascheid, “Automatic detection of concurrency bugs through event ordering constraints”, *Design, Automation and Test in Europe Conference and Exhibition (DATE'14)*, pp. 1-6, 2014