Simulation and estimation for MPSoC programming tools

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

Rapido Workshop. HiPEAC Conference
Amsterdam, January 21st 2015

Acknowledgements

- German Cluster of Excellence: Center for Advancing Electronics Dresden (**www.cfaed.tu-dresden.de**)
- Silexica Software Solutions GmbH
- Institute for Communication Technologies and Embedded Systems (ICE), RWTH Aachen: Prof. Leupers

Agenda

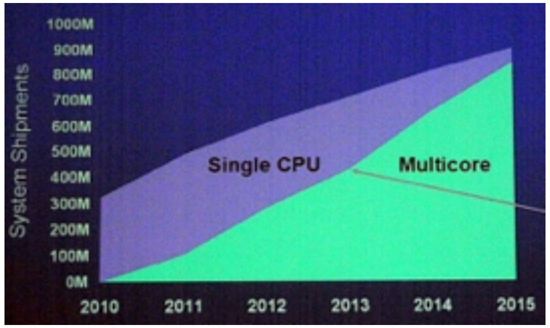
- MPSoC compilation
- Simulation & estimation for timing information
- Simulation: other use-cases

Agenda

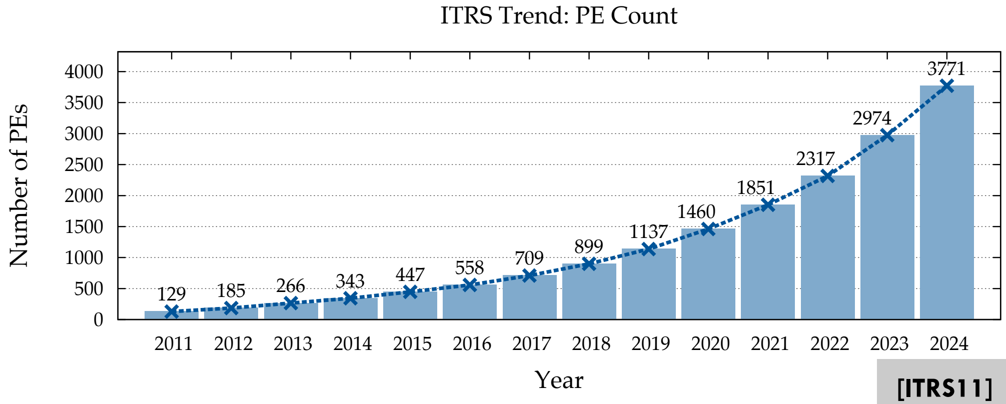
- **MPSoC compilation**
- Simulation & estimation for timing information
- Simulation: other use-cases

Multi-Processor on Systems on Chip (MPSoCs)

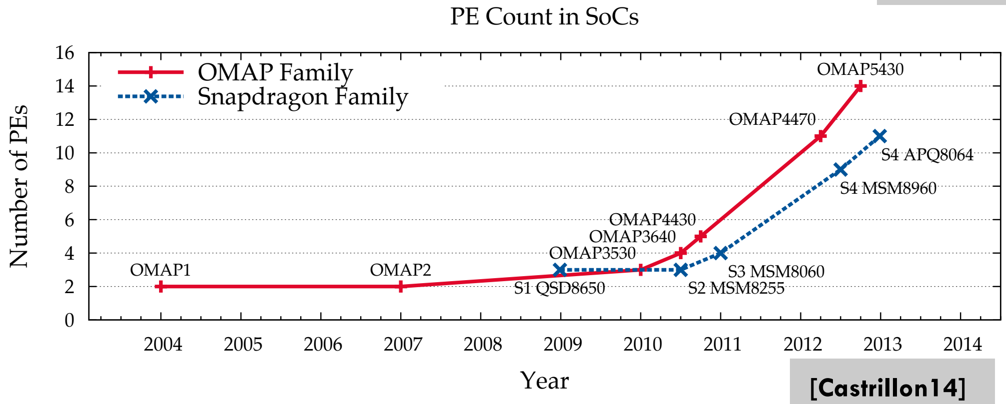
- **HW complexity**
  - Increasing number of cores
  - Increasing heterogeneity
- Multi-cores everywhere
  - Ex.: Smartphones, tablets and e-readers



[EETimes11]



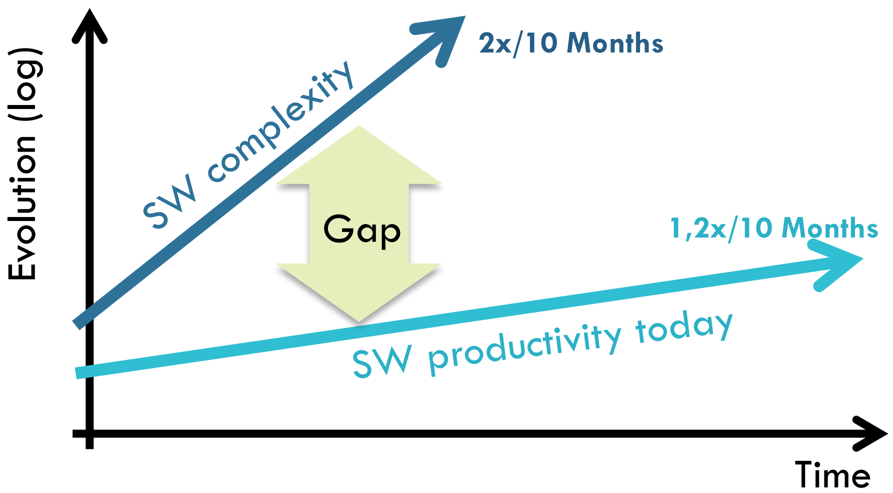
[ITRS11]



[Castrillon14]

MPSoCs: SW productivity gap

- SW-productivity gap: complex SW for ever-increasing complex HW
  - → Cannot keep pace with requirements
  - → Cannot leverage available parallelism



Source: Qualcomm, Texas Instruments

MAPS MPSoC Compiler

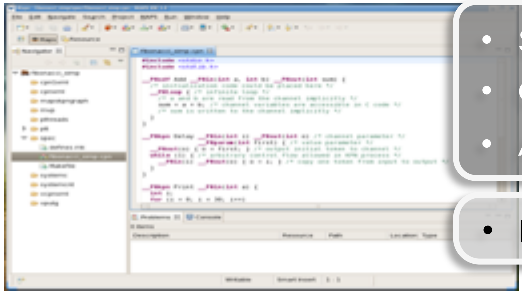
Application

- Sequential C legacy code
- Parallel KPN code

Architecture model

- PE, multi-tasking and communication APIs

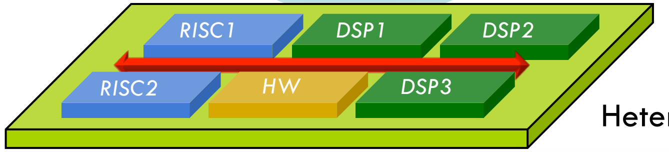
Eclipse integration



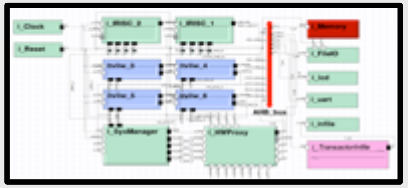
- Sequential/parallel code profiling & tracing
- Code partitioning / Mapping and scheduling
- Automatic target C code generation

SW performance estimation or measurement on real HW

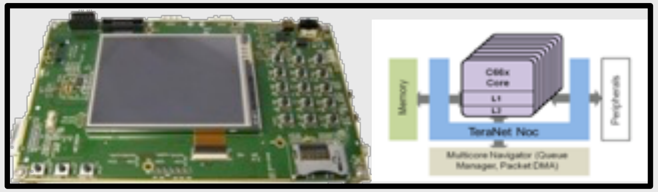
- Native C compilers for PEs (vendor compilers)



Heterogeneous MPSoC



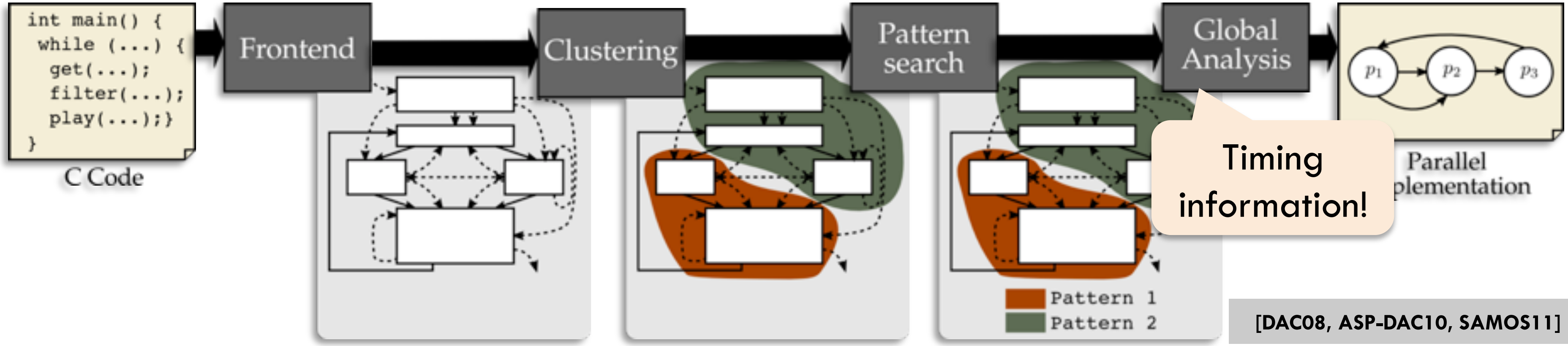
VP (ARM9, VLIW, RISC)



TI SoCs: OMAP, Keystone

Tegra 3 tablet

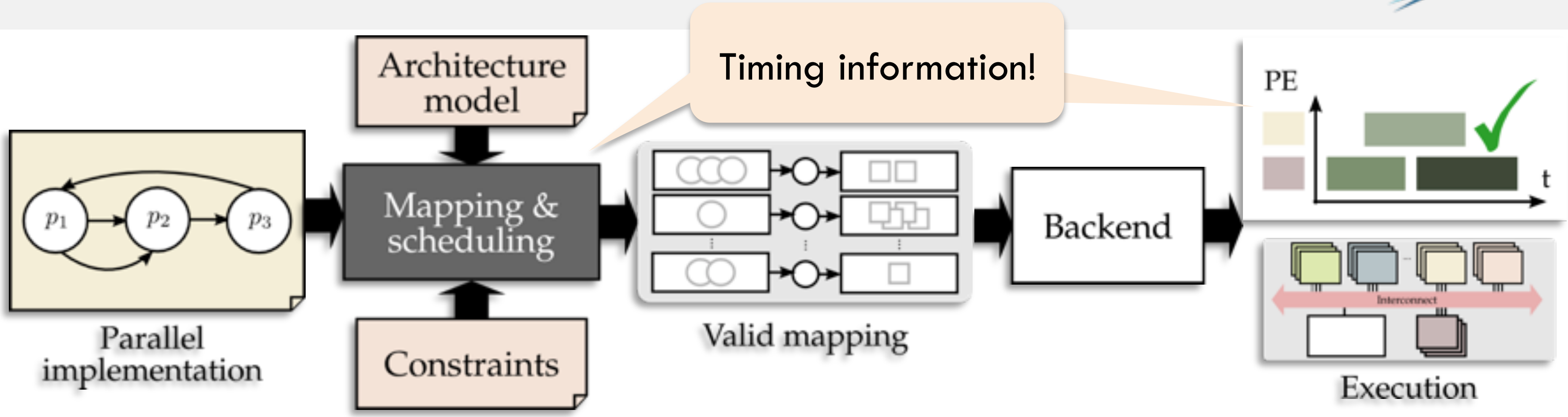
Handling sequential C code



[DAC08, ASP-DAC10, SAMOS11]

- Purpose: Expose parallelism from a sequential specification
  - Frontend: Build graph model of the application (dynamic DFA)
  - Clustering: Group statements into potential parallel tasks
  - Pattern search: Annotate graph with parallelism patterns
  - Global analysis: Fix final configuration (➔ implementation)

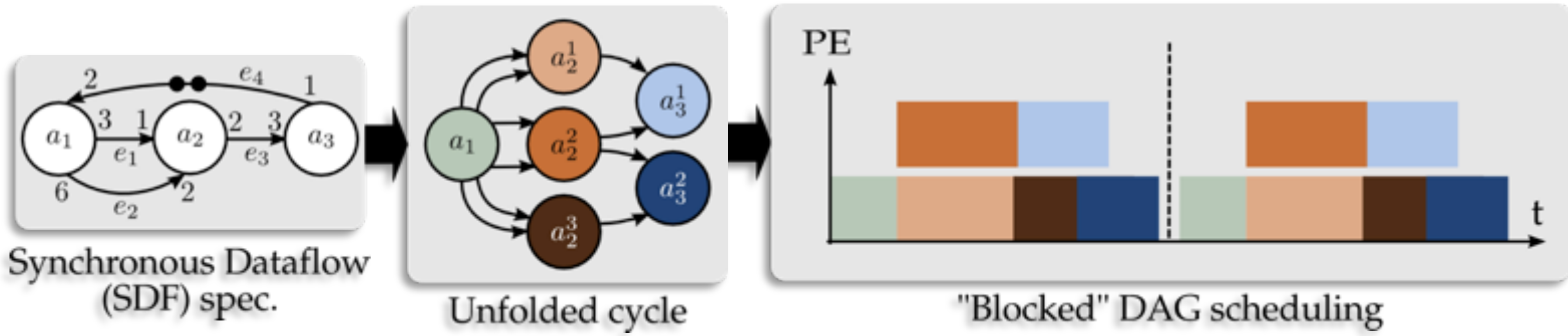
Handling parallel code



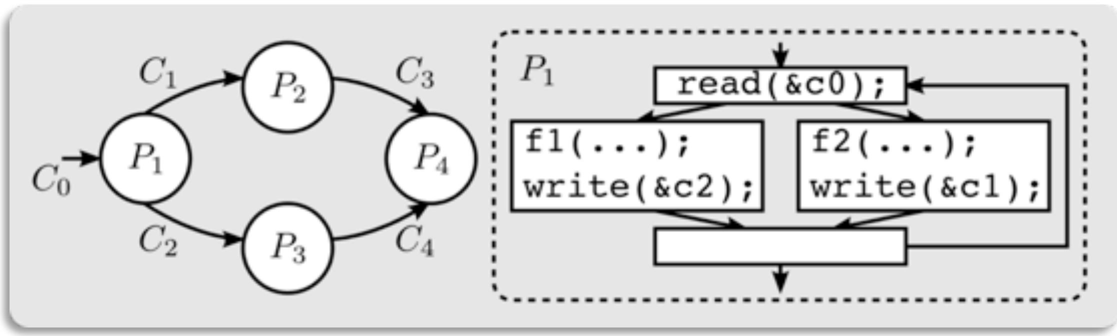
- Input: Kahn Process Networks (KPN) & other flavors of dataflow models
  - A node (process) represents computation
  - An edge (channel) represents communication
- Output: Valid **heterogeneous mapping** (➔ comply to constraints)
  - Process and channel mapping
  - Buffer sizing: Memory allocated for communication

Dataflow models: static vs. dynamic

- Static: Synchronous dataflow models



- KPNs (& DDF)
  - No “hardcoded” rates
  - More expressiveness
  - More difficult to analyze



C Extension for KPNs

- FIFO Channels

```
typedef struct { int i; double d; } my_struct_t;
__PNchannel my_struct_t C;
__PNchannel int A = {1, 2, 3}; /* Initialization */
__PNchannel short C[2], D[2], F[2], G[2];
```

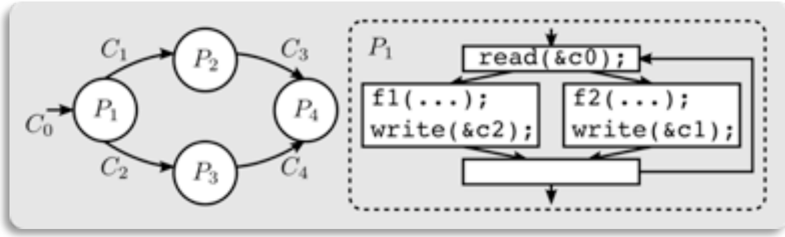
- Processes & networks

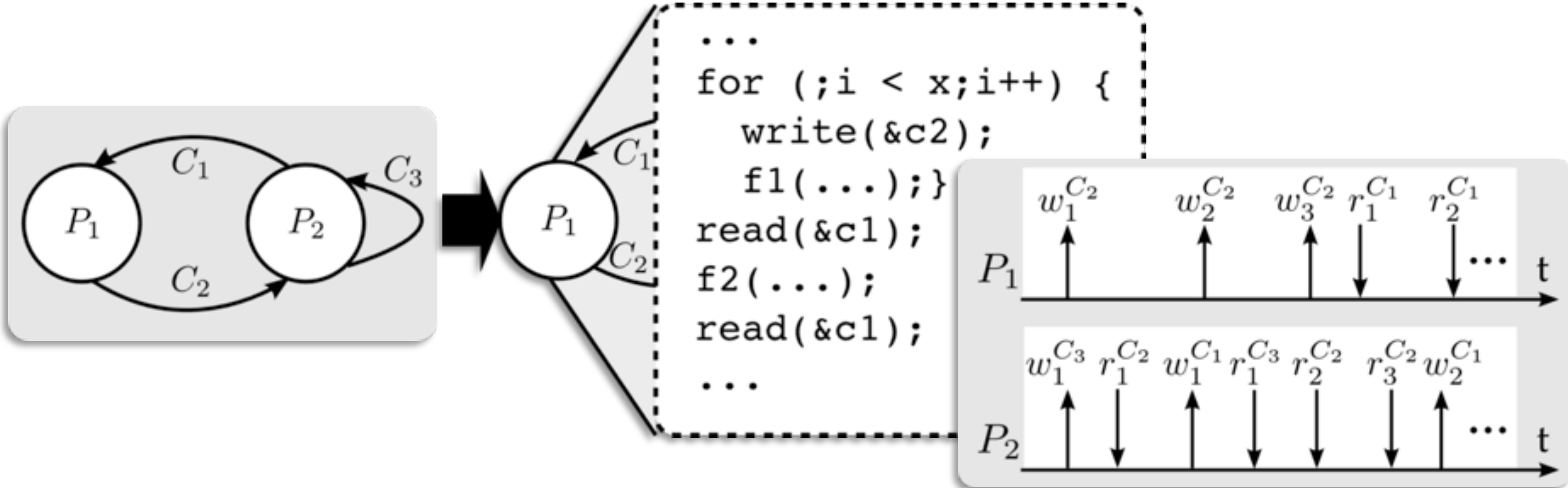
```
__PNkpn AudioAmp __PNin(short A[2]) __PNout(short B[2])
                 __PNparam(short boost){
  while (1)
    __PNin(A) __PNout(B) {
      for (int i = 0; i < 2; i++)
        B[i] = A[i]*boost;
    }}
__PNprocess Amp1 = AudioAmp __PNin(C) __PNout(F) __PNparam(3);
__PNprocess Amp2 = AudioAmp __PNin(D)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

[PARCO14]

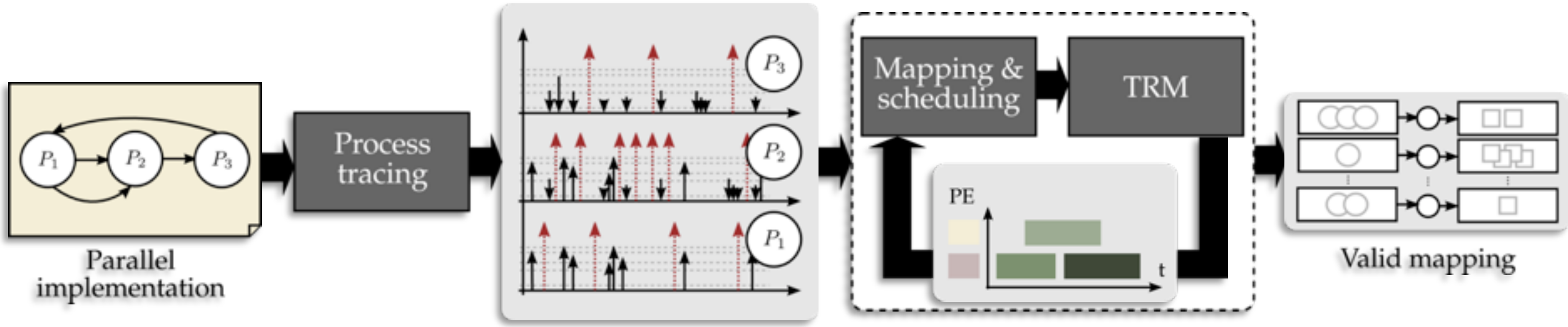
Dealing with dynamic behavior: tracing

- White model of processes: source code analysis and tracing





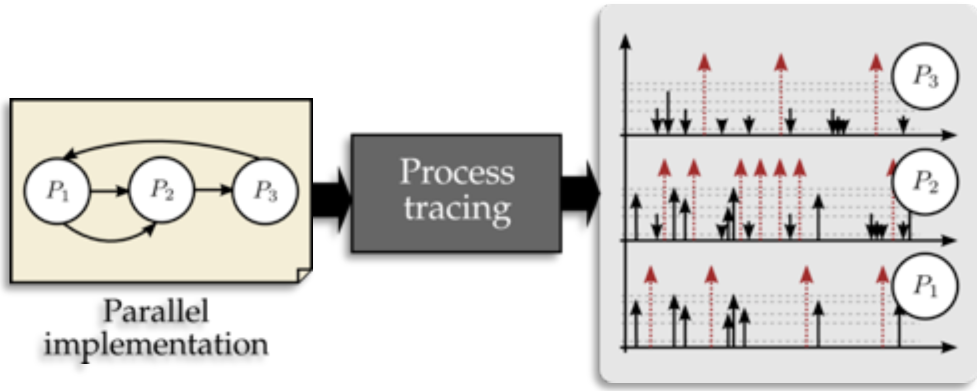
Trace-based mapping and scheduling

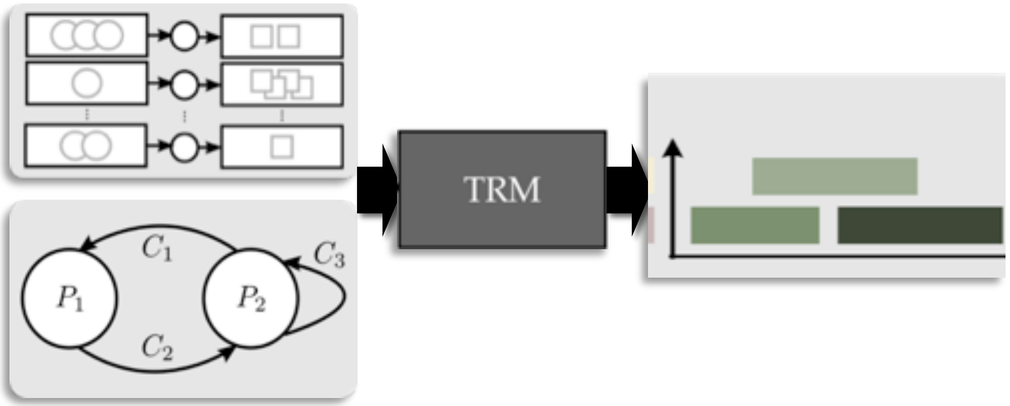


- Mapping: Trace-based heuristics
  - Mapping & scheduling: Analyze traces and propose mapping
  - Iterate: Improve mapping (if required)

[DATE10, DAC12, IEEE-TII13]

Obtaining/generating timing information





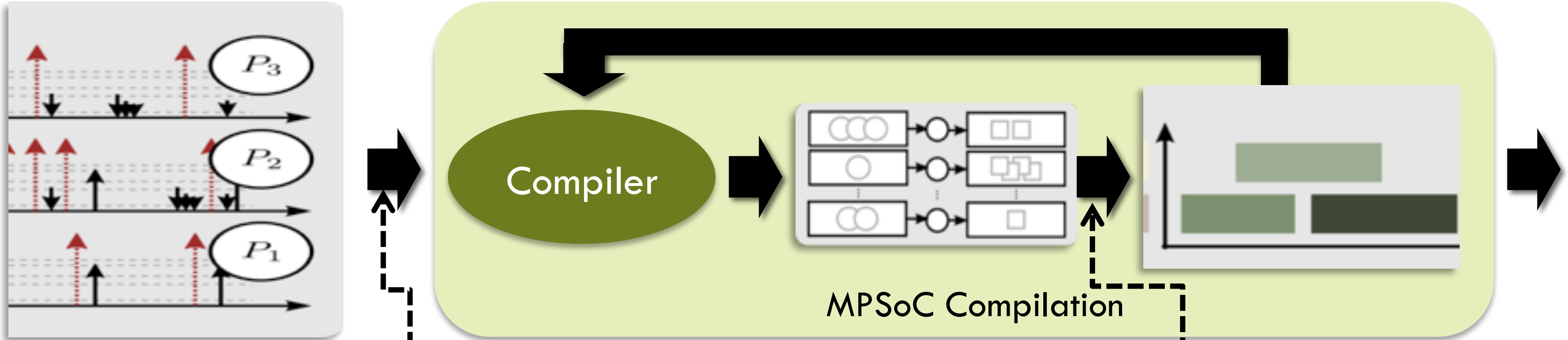
- Computation time elapsed between events
  - For different processor types
  - **Fine-grained information**

- Parallel execution: Models for
  - OS
  - Communication
  - Task management and synchronization

Agenda

- MPSoC compilation
- **Simulation & estimation for timing information**
- Simulation: other use-cases

Timing information for MPSoC compilation



Sequential performance estimation

Design ↓

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

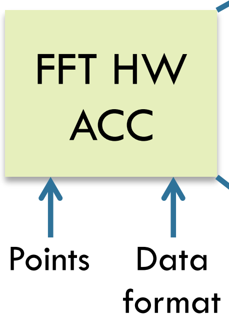
Parallel performance estimation

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

Cost-tables and annotations

- Cost tables
  - Equivalence: Low-level IR → Assembly instructions
  - Coarse estimation of instruction-level parallelism

- Annotation: Coarser and parametrizable [SDR10, ALOG11]
  - Datasheet parametrizable equations for hardware accelerators

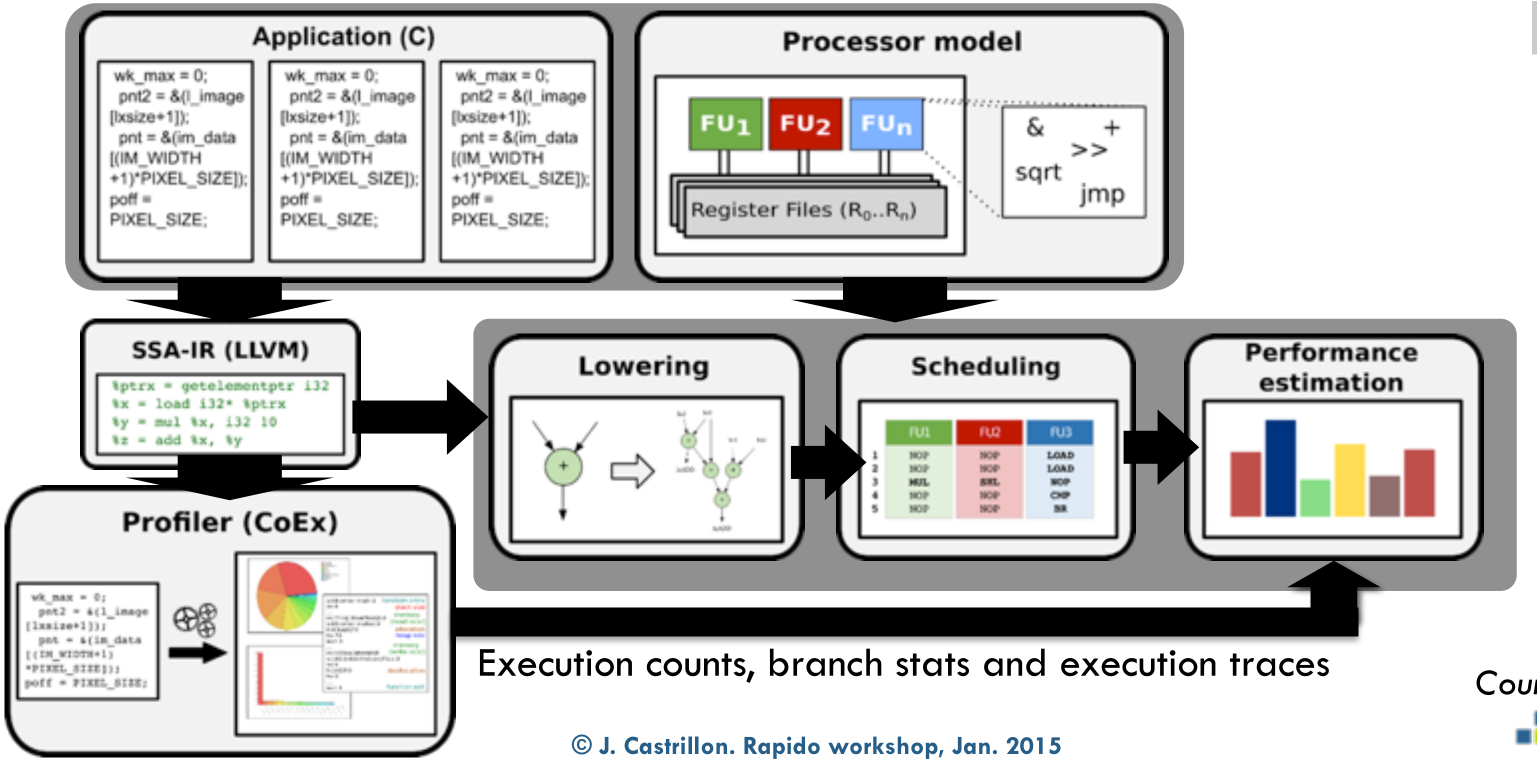



```
avor name="fft_HW" Nucleus="FFT">
arameter name="bitwidth"><value>32.0</value></parameter>
arameter name="points"><values List="32 64 128 256" /></para
roperty name="latency"><function>16*points+100</function></p
nput name="fft_in"><port>input</port>
DataType representation="fixed_point" format="Q31" DataWidth
Interface type="buffer_flag_1of2">
<val name="size" val="8" /><val name="stride" /> <val name="
<val name="fsize" val="4" /></Interface>
Interface type="buffer_flag_2of2">
<addr name="addr" pool="in" min="0x04100000" max="0x041F0000
```

Processor performance models

- Architecture description languages (e.g., LISA) & abstract processor models

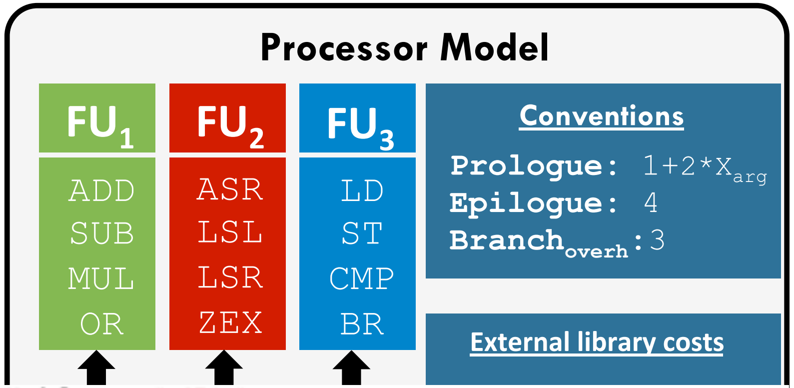
[SAMOS14]



Courtesy: J. Eusse

Processor performance models (2)

- Abstract models for compiler emulation
  - Set of resources (functional units, register banks)
  - Set of operations (pipeline effects, SIMD, addressing modes, predicated execution)
  - SW-related costs (calling convention, register spilling, C-lib calls)

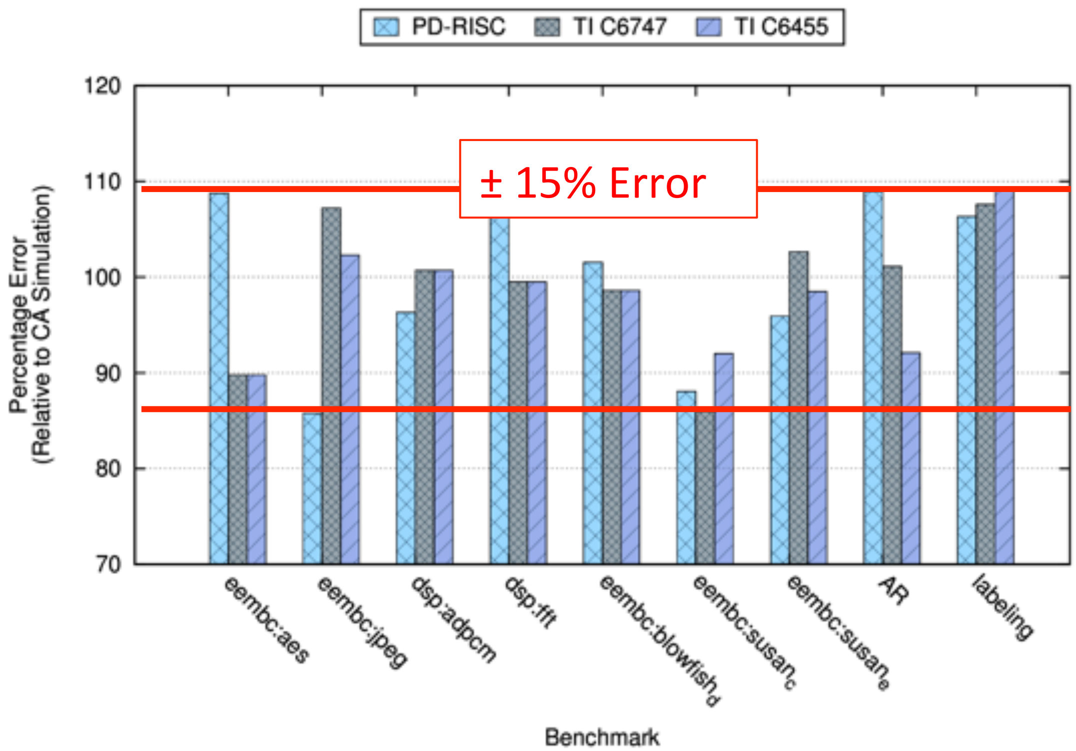



```
<Model name="c674x"> ...
  <FunctionalUnit BW="32" Ctrl="0" name="FU1">
    <Operation bw="32" Pipe="1" lat="2" name="MUL"
            Imm="16" SIMD="1"/>
    <Operation bw="16" Pipe="0" lat="2" name="ADD"
            Imm="16" SIMD="2"/>
  </FunctionalUnit>
  ...
  <RegisterFile name="RF1" size="16" BW="32"/>
  <RegisterFile name="RF2" size="8"  BW="32"/>
  <Perilogue proBase="1" proLin="2" epBase="4" epiLin="0"/>
  <LibraryCosts >
    <Cost name="malloc" base="9"   lin="0.3"/>
    <Cost name="fsqrt"  base="235" lin="0" />
    ...
  </LibraryCosts>
</Model>
```

Courtesy: J. Eusse

Processor performance models: Results

[SAMOS14]

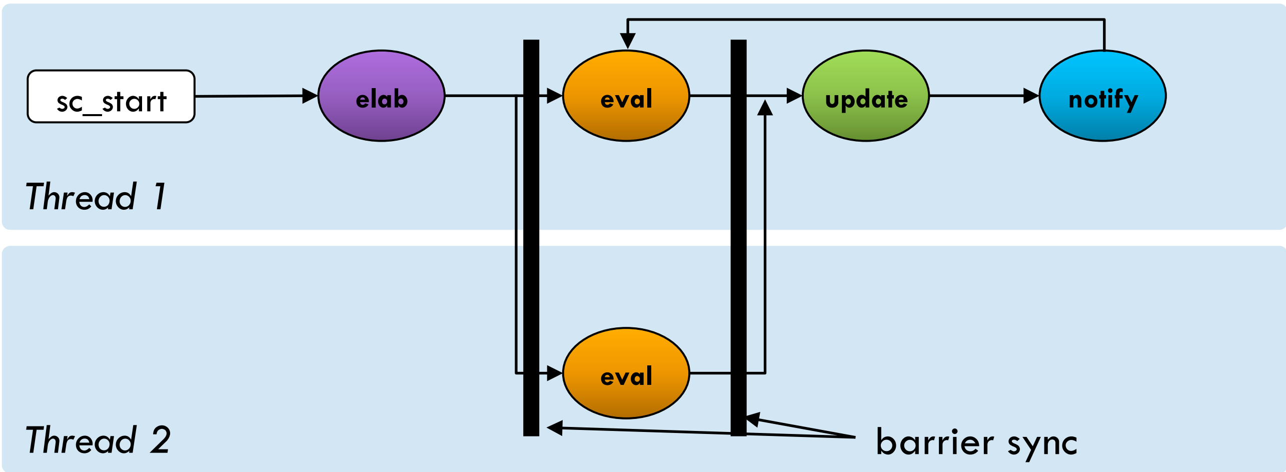


Average gain:
248x (PD-RISC)
67x (TI DSPs)
(CA vs. profiling + estimation time)

Courtesy: J. Eusse

Speeding-up system simulators

- Research on instruction set simulators: Interpretative, compiled, just-in-time compiled, dynamic binary translation, …
- System simulators (i.e. **Virtual Platforms**): parallel SystemC
  - **ParSC:** conservative, synchronous simulation (delta-cycle), good for **cycle-accurate** simulation



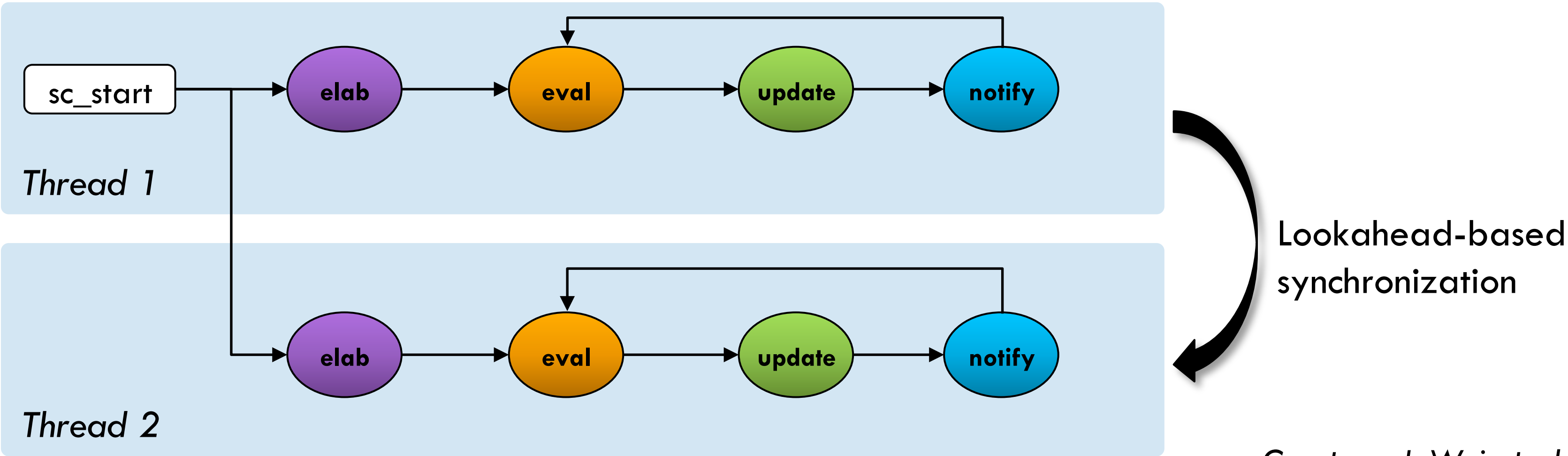
[CODES/ISSS10]

Courtesy: J. Weinstock
R. Leupers

Speeding-up system simulators (2)

[DATE14a]

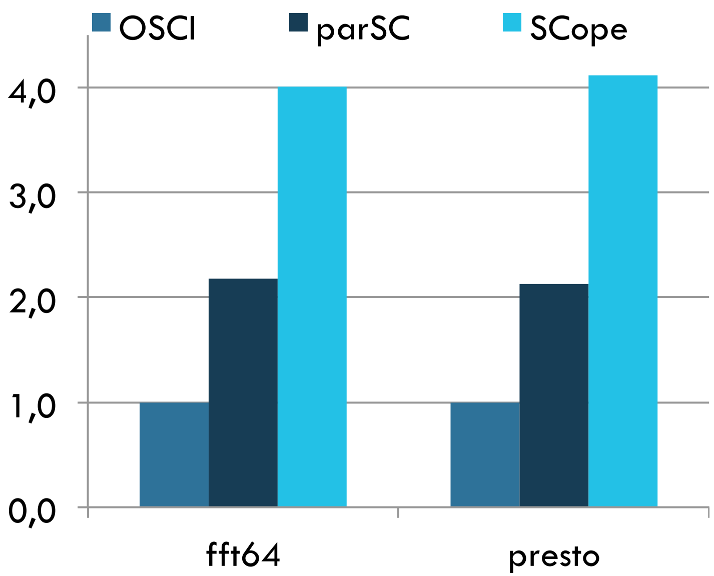
- System simulators: parallel SystemC (Cont.)
  - **SCope:** conservative, time-decoupled simulation (quantum-based), good for TLM and instruction-accurate simulation



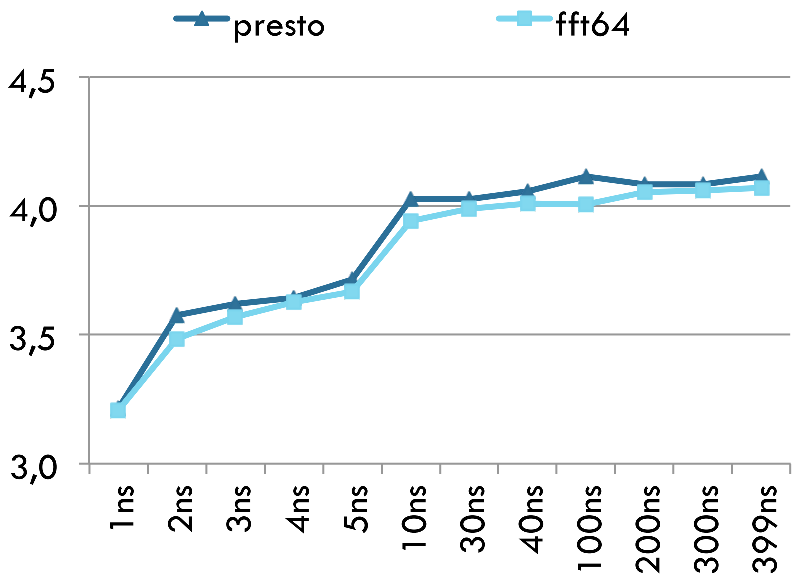
Courtesy: J. Weinstock
R. Leupers

System simulation: speedup

1. Speedup vs. plain SystemC



2. Speedup vs. Lookahead



Simulation host: Quad-Core simulation host (Intel i7 920), 4 threads

Courtesy: L. Murillo, J. Weinstock
EURETILE EU Project

Agenda

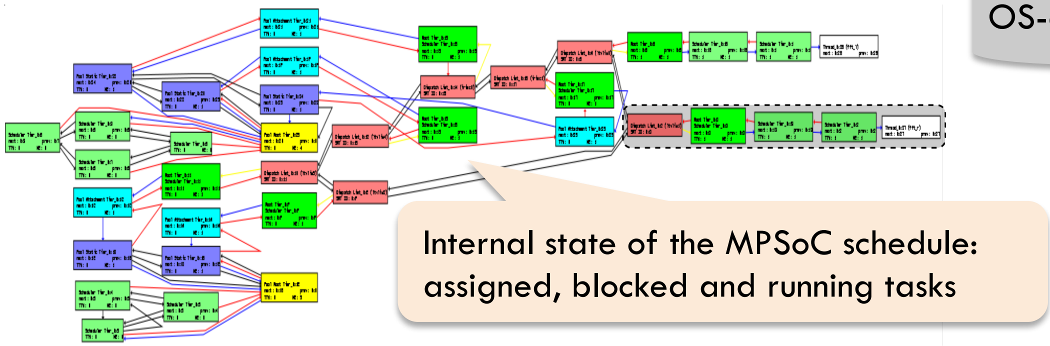
- MPSoC compilation
- Simulation & estimation for timing information
- **Simulation: other use-cases**

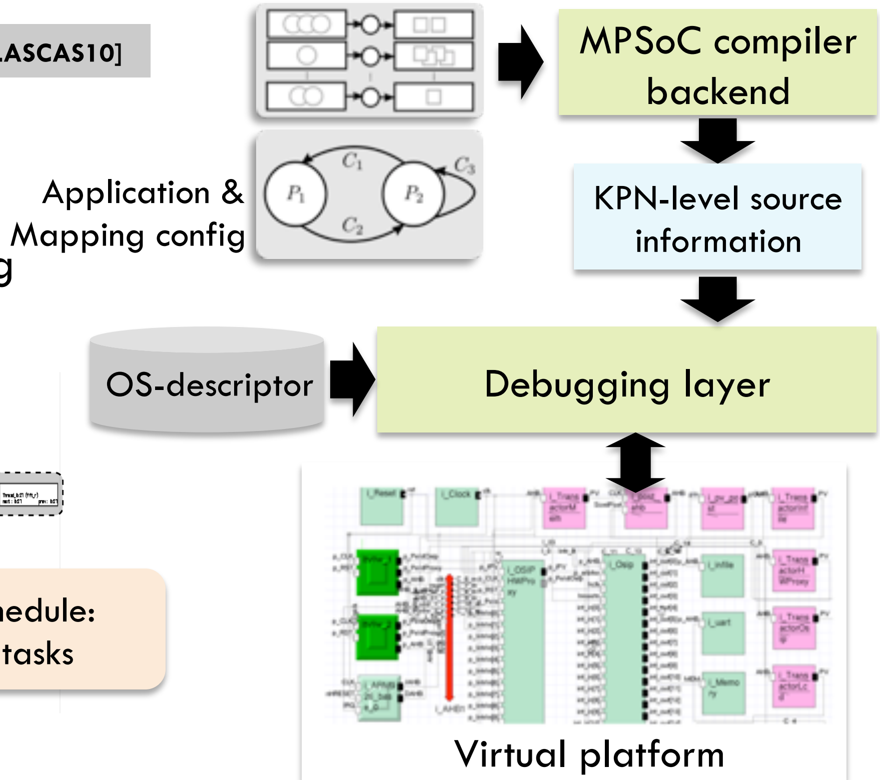
Virtual platforms: use cases

- Debugging: exploit observability and controllability
- Power/energy estimation: exploit abstraction

Debugging with virtual platforms

- Interactive debugging [LASCAS10]
  - Get snapshots of the system state
  - Full system stop
  - Track progress irrespective of mapping

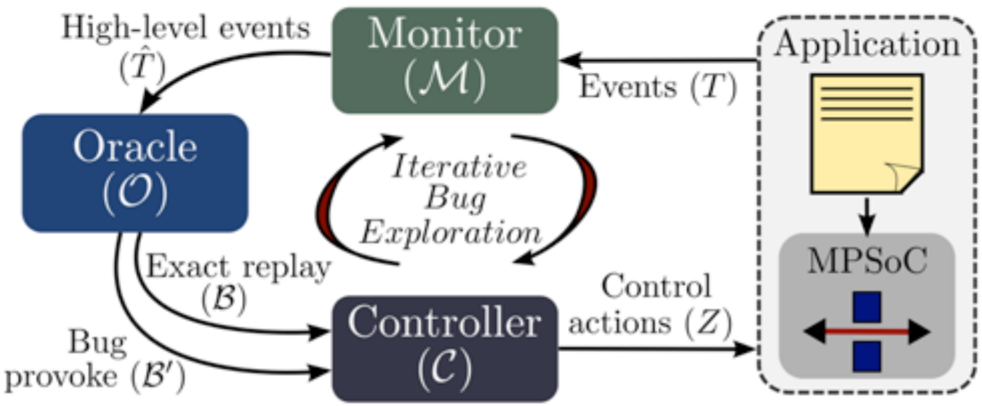




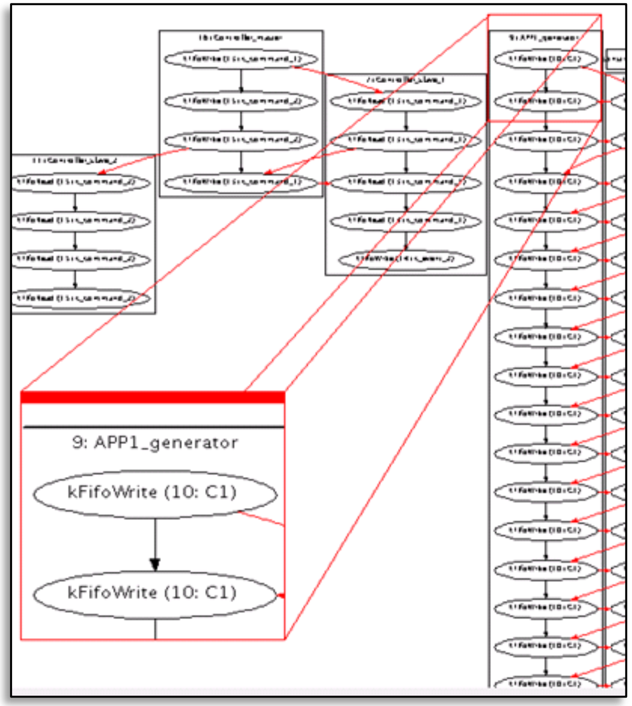
Debugging with virtual platforms (2)

[DATE14b]

- Deterministic replay and automatic bug exploration



Courtesy: L. Murillo

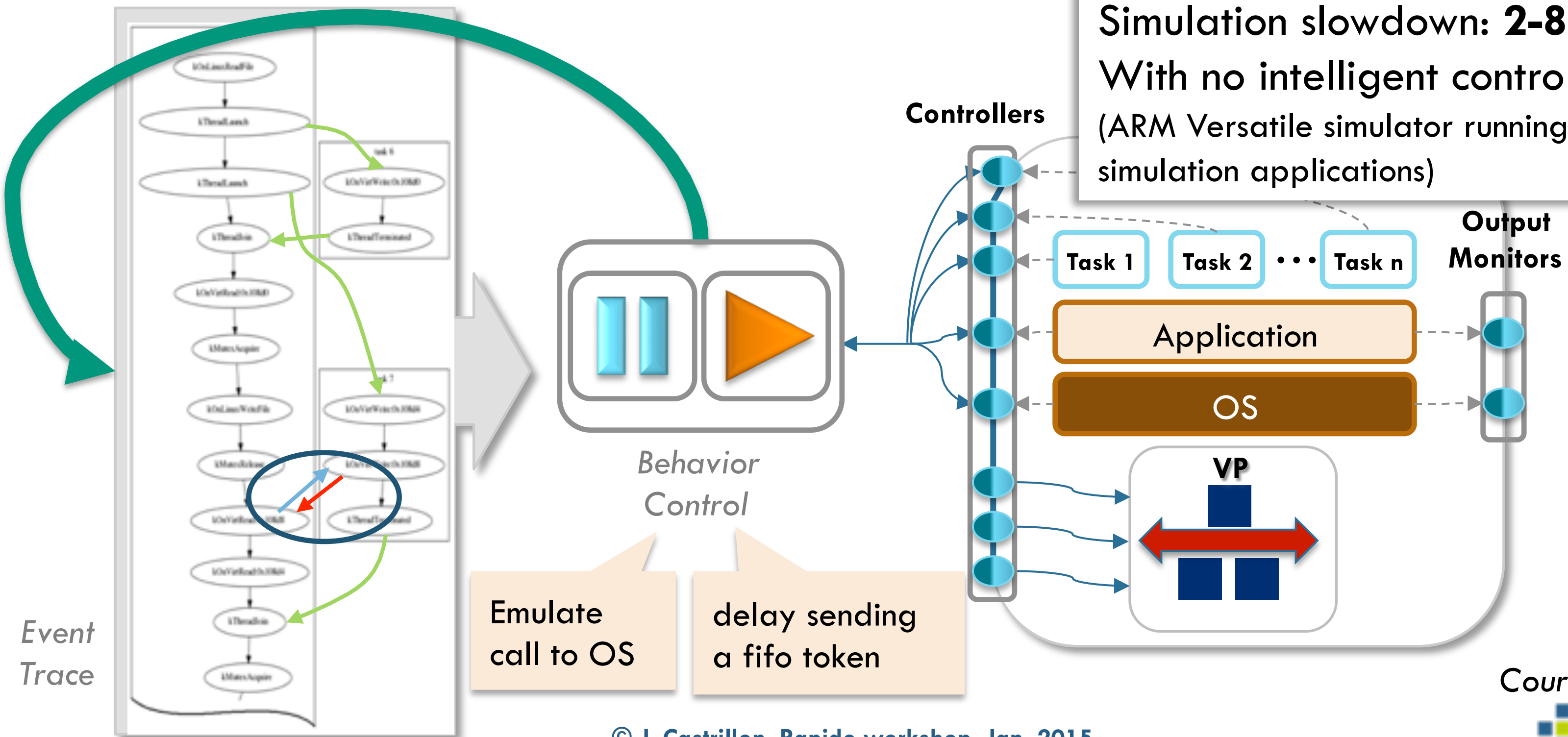


Debugging with virtual platforms (3)

- Controlling and swapping event orders

[DATE14b]

Simulation slowdown: **2-80x**
With no intelligent control: **~1600x**
(ARM Versatile simulator running an ocean simulation applications)

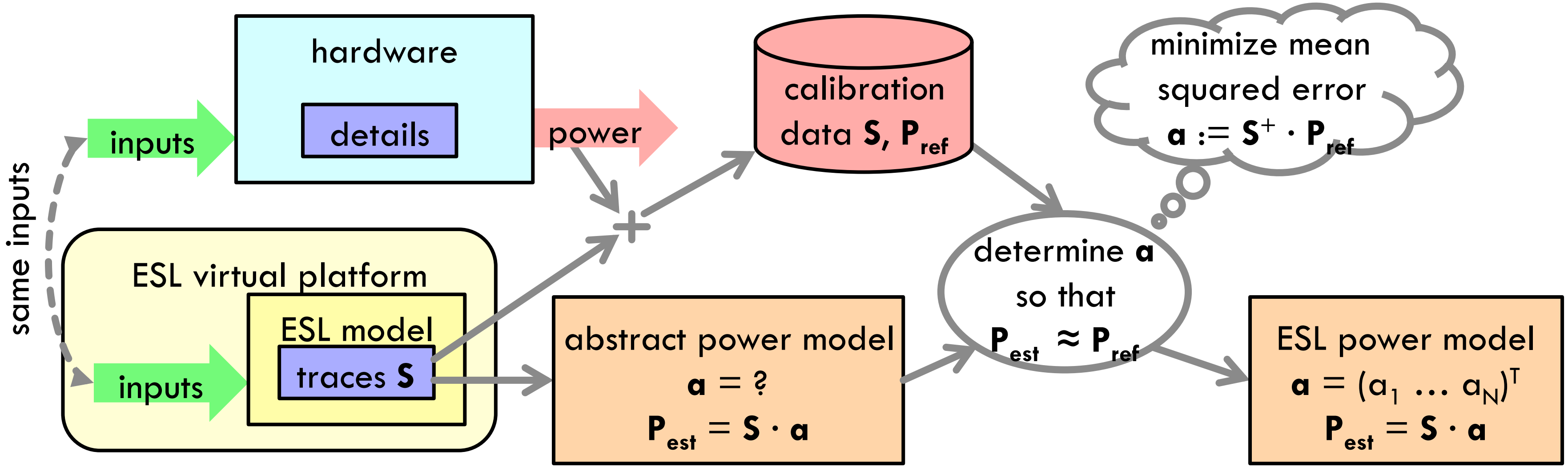


Courtesy: L. Murillo

ESL power estimation

- Callibrate abstract power models from hardware measurements
  - Run HW and ESL models with same input
  - Record traces for calibration

[DAC13]

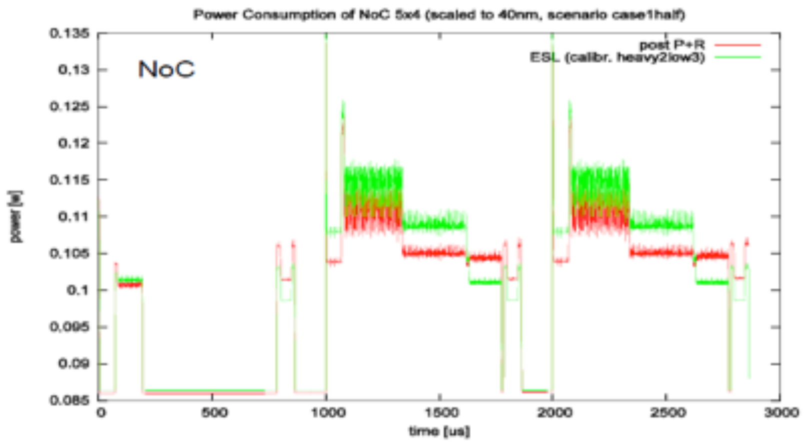


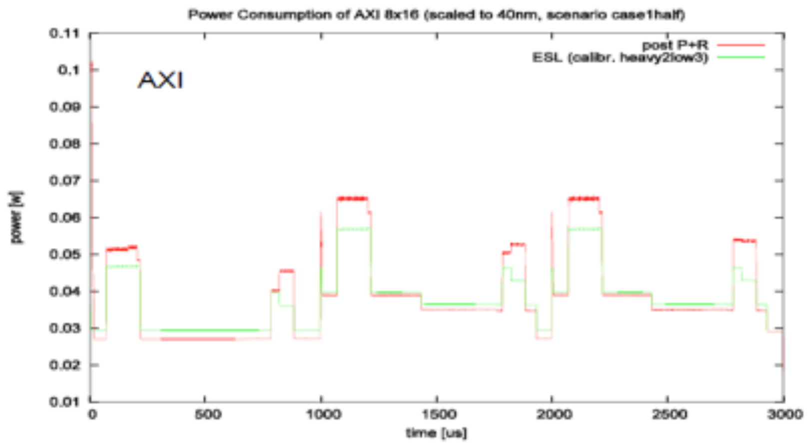
Courtesy: S. Schürmans, R. Leupers

ESL power estimation (2)

[DAC13]

- Evaluation: AMBA AXI and Custom NoC design (post P&R)
- Results
  - **Error < 22%** (accurate prediction of phases)
  - **Speedup 900x (**vs. low-level power simulation)





Courtesy: S. Schürmans, R. Leupers

Summary

- Discussed academic (and commercial) MPSoC programming tools

- The role of simulation/emulation technology for MPSoC compilers
  - Different technologies for different design phases
  - Deal with heterogeneous architectures
  - Deal with system-level simulators

- New, important use-cases for virtual platforms
  - Debugging
  - Power/energy estimation

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

[EETimes11] http://www.eetimes.com/document.asp?doc_id=1259446

[DAC08] J. Ceng, J. Castrillon, W. Sheng, H. Scharwächter, R. Leupers, G. Ascheid, H. Meyr, T. Isshiki, and H. Kunieda, “MAPS: An integrated framework for MPSoC application parallelization,” in Proceedings of the Design Automation Conference, 2008

[ASP-DAC10] R. Leupers and J. Castrillon, “MPSoC programming using the MAPS compiler,” in Proceedings of the Asia and South Pacific Design Automation Conference, 2010

[SAMOS11] J. Castrillon, W. Sheng, and R. Leupers, “Trends in embedded software synthesis,” in Proceedings of the International Conference on Embedded Computer Systems: Architecture, Modeling and Simulation, 2011

[PARCO14] W. Sheng, S. Schürmans, M. Odendahl, M. Bertsch, V. Volevach, R. Leupers, and G. Ascheid, “A compiler infrastructure for embedded heterogeneous MPSoCs”, Parallel Comput. 40, 2 (February 2014), 51-68

[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

[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

References (2)

[**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

[**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

[**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

[**CODES/ISSS10**] C. Schumacher, R. Leupers, D. Petras, A. Hoffmann, “parSC: synchronous parallel systemc simulation on multi-core host architectures”, Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, CODES/ISSS’10. pp. 241-246, 2010

[**DATE14a**] J. H. Weinstock, C. Schumacher, R. Leupers, G. Ascheid, L. Tosoratto, “Time-decoupled parallel SystemC simulation”, Proceedings of the conference on Design, Automation & Test in Europe (DATE’14), 2014

[**LASCAS10**] 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

[**DATE14b**] 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

[**DAC13**] S. Schürmans, D. Zhang, D. Auras, R. Leupers, G. Ascheid, X. Chen, and L. Wang, “Creation of ESL Power Models for Communication Architectures using Automatic Calibration”, Proceedings of the 50th Annual Design Automation Conference (DAC’13), 2013