A long and winding road towards predictability...

Eric JENN - IRT Saint-Exupéry

Agenda.

1. The context and the problems
2. Determinism by analysis
3. Determinism by design
4. Conclusion

The context

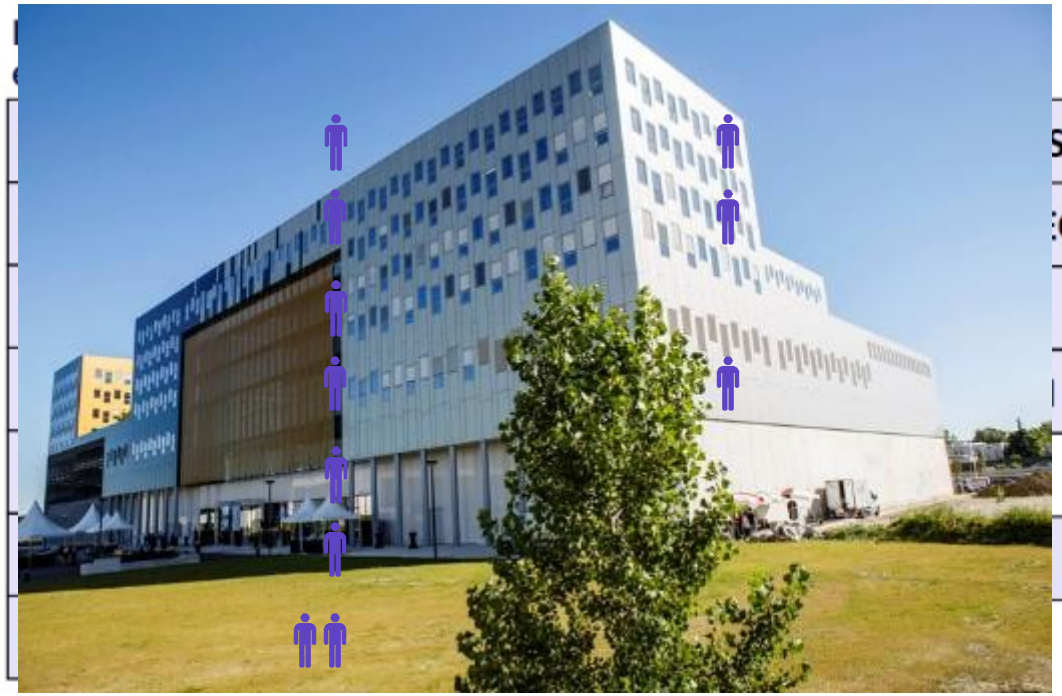
Who are we?

The context

IRT Saint-Exupery in Technogical Research Institute

- Focus on aeronautic, space, automotive
- Covers aspect related to materials, electrical systems, computing system, communication, artificial intelligence
- Projects co-funded by industry
  - Strongly driven by industrial needs
  - Focused on technological transfer (TRL 4-5, sometimes lower…)
- Work carried out by a composite team of engineers (seconded by their companies), academic researchers, post-docs

CAPHCA project



The problem

What was the question?

The problem

Dependability and performance

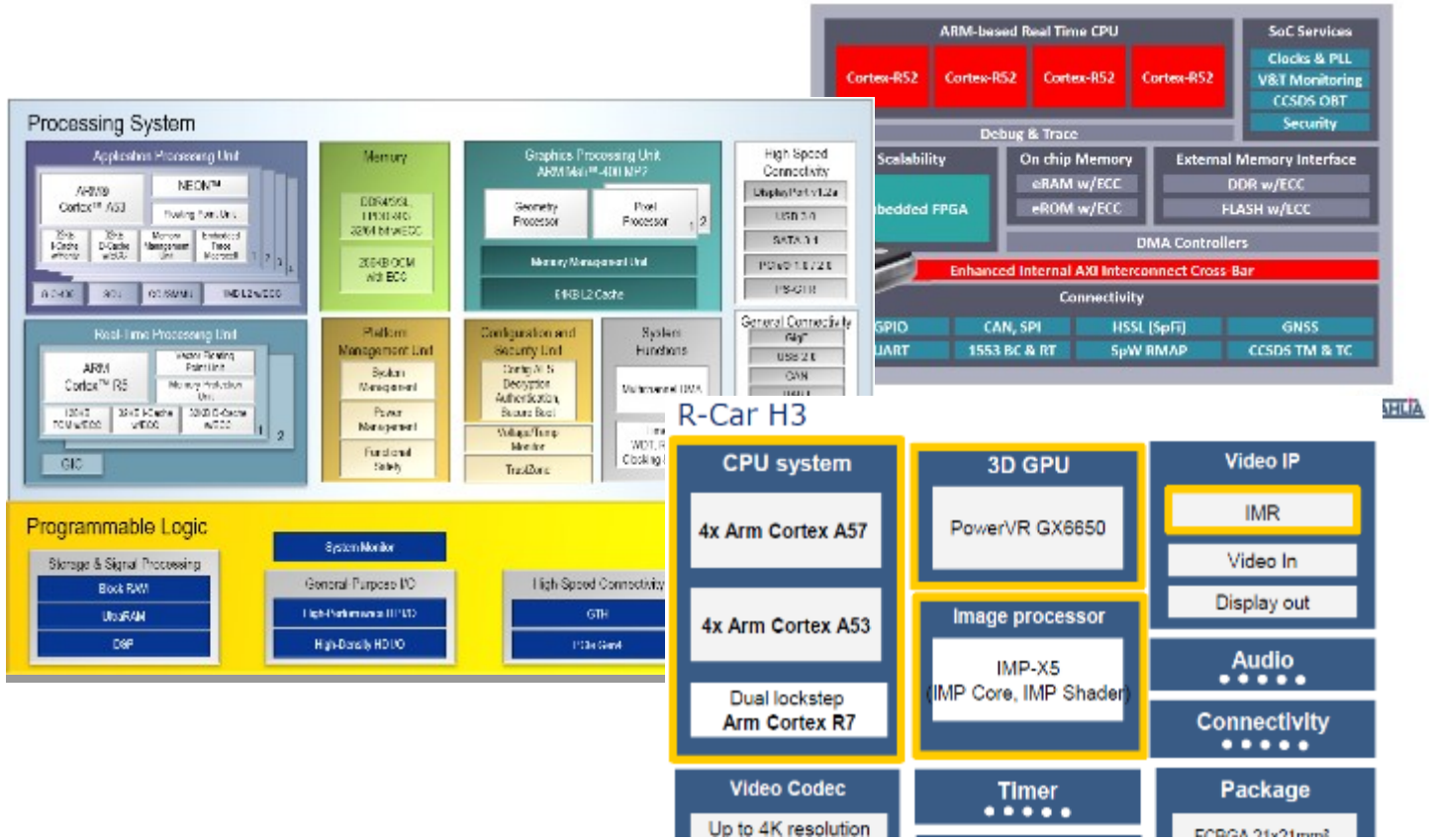
- **Dependability:** the extent to which **confidence** can be placed on the capability of the system to fulfil its intended purpose
- **Performance:** the **efficiency** with which a system fulfils its intended purpose

The problem

Emergence of new computation platforms

- Multiple cores (multi + many)
  - Complex cores
  - Heterogeneous cores
- SIMD units
- GPU
- FPGA
- AI accelerators

- Interconnect
- SDRAM
- …



Source RENESAS

Master complexity

Compliance with certification constraints

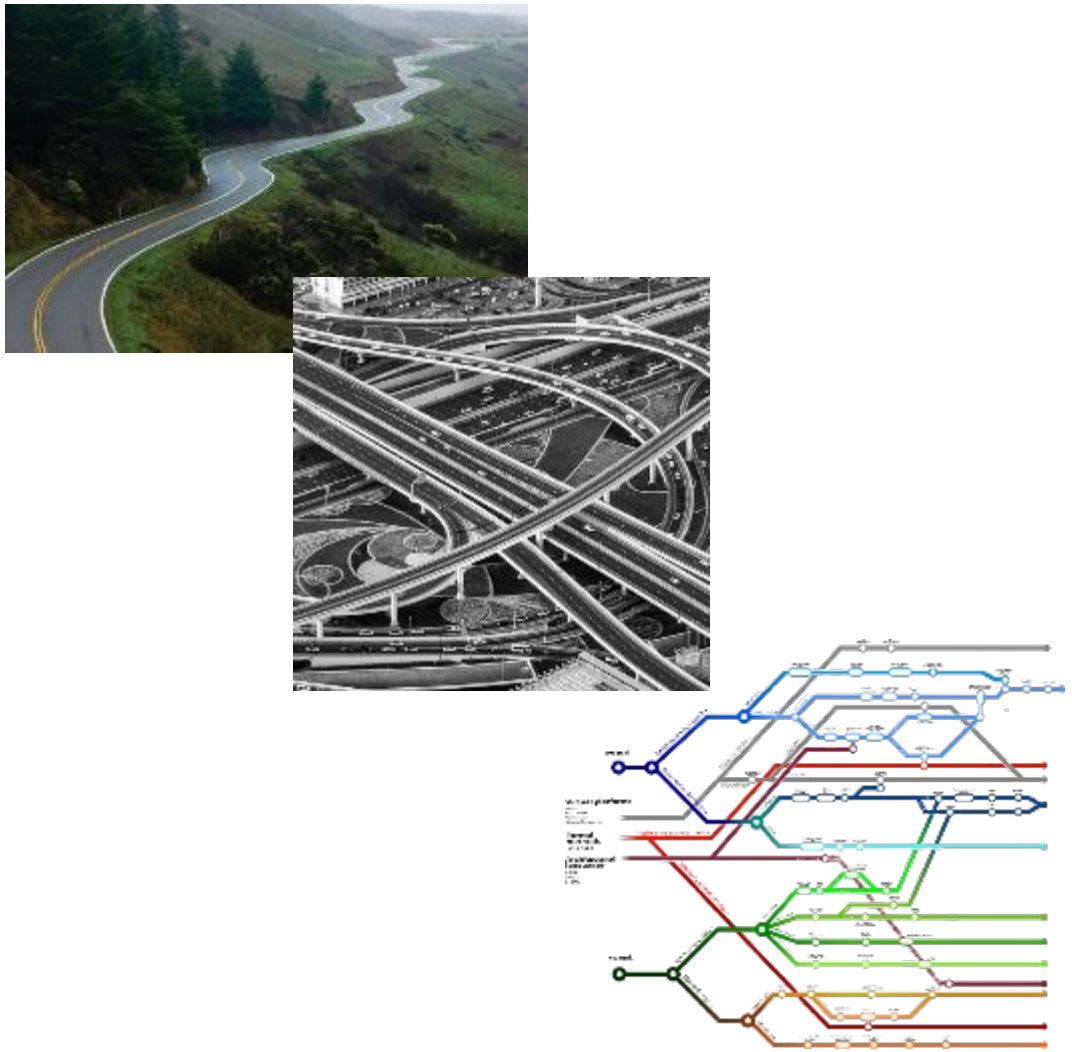
Use platforms efficiently (reduce margins)

Reduce development (incl. V&V) costs

A long and winding road...

How to...

- ensure determinism and predictability?
  - Live with variability ?
  - Reduce variability?
- chose effective / reasonable (cost effective) solution?



Predictability by analysis

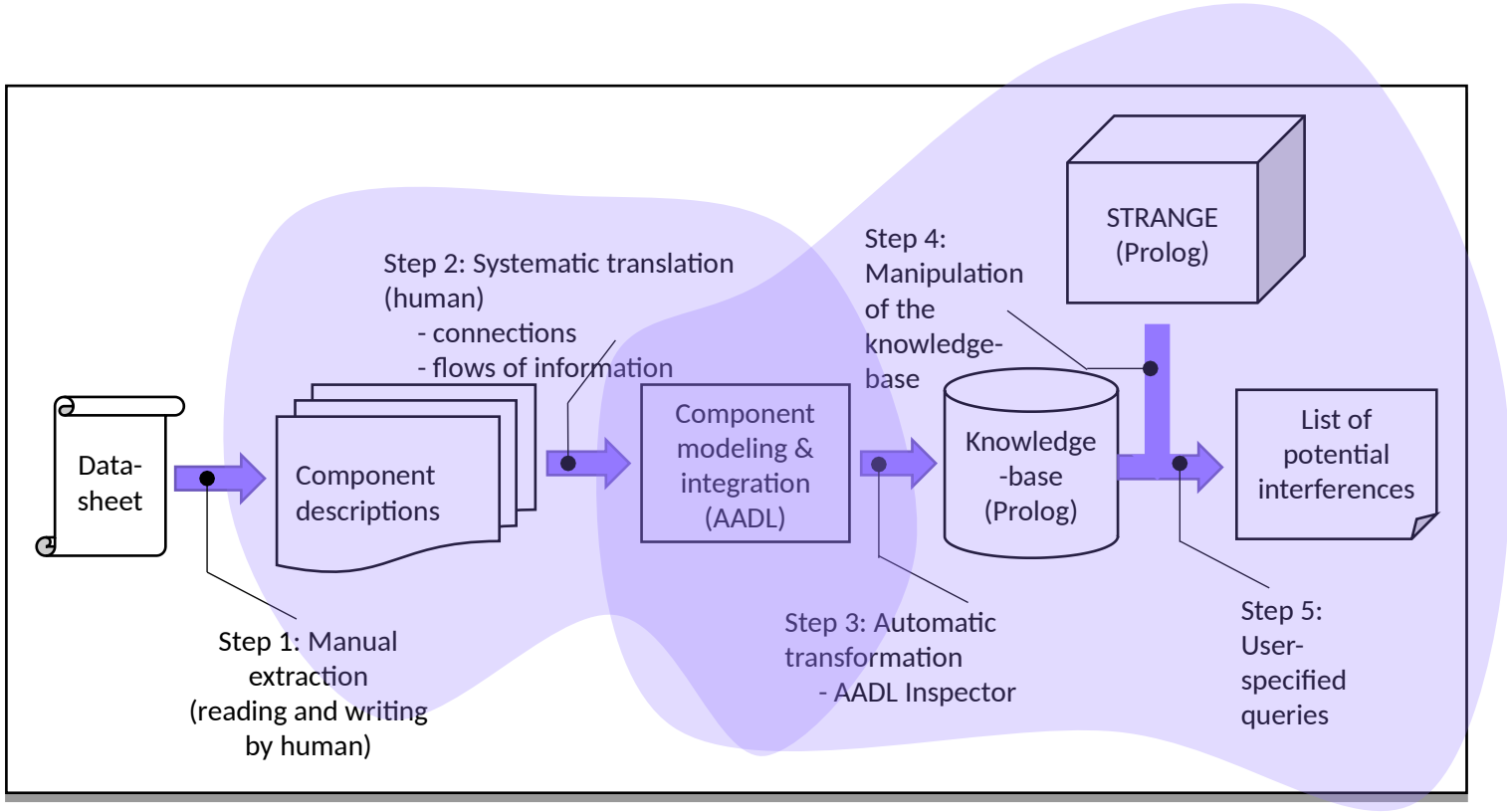
How to model and analyse a platform?

Identifying interference sources

Modeling for interference analysis

- Processeurs are (very) complex
  - TC277, more than 5000 pages
- Documentation is developer-oriented
- Documentation is not always correct / complete

- **How to ensure the completeness of the analysis?**



Identifying interference sources

Modeling for interference analysis

- Processeurs are (very) complex
  - TC277, more than 5000 pages
- Documentation is developer-oriented
- Documentation is not always correct / complete

- **How to ensure the completeness of the analysis?**
  - Identify “components”, flows of transactions
  - Capture elements using AADL

CD

Hosting component: Top-level
Interfaces
- Connect to A, C, E
Information flows
- *Request received* from: A, C
- *Request sent/forwarded* to: E (from A and C)
- Data sent/received: as results of requests

translate

AADL

```
system D
   features
        d1: in out event data port; -- to A
        d2: in out event data port; -- to C
        d3: in out event data port; -- to E
   flows
        d_f1: flow path d1 -> d3; -- A to E
        d_f2: flow path d2 -> d3; -- C to E
end D;

system implementation D.imp
   subcomponents
        f : system F;
   connections
        d1_f1: port d1 <-> f.f1;
        d2_f1: port d2 <-> f.f1;
        d3_f2: port d3 <-> f.f2;
   flows
        d_f1 : flow path d1 -> d1_f1 -> f.f_f -> d3_f2 -> d3;
        d_f2 : flow path d2 -> d2_f1 -> f.f_f -> d3_f2 -> d3;
end D.imp;
```

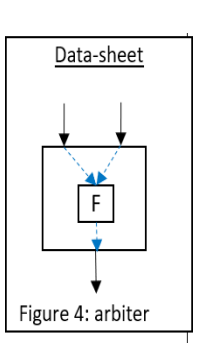


Figure 4: arbiter

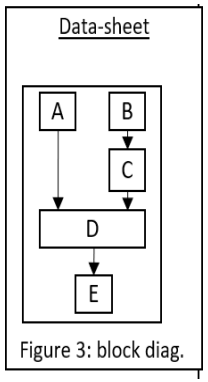


Figure 3: block diag.

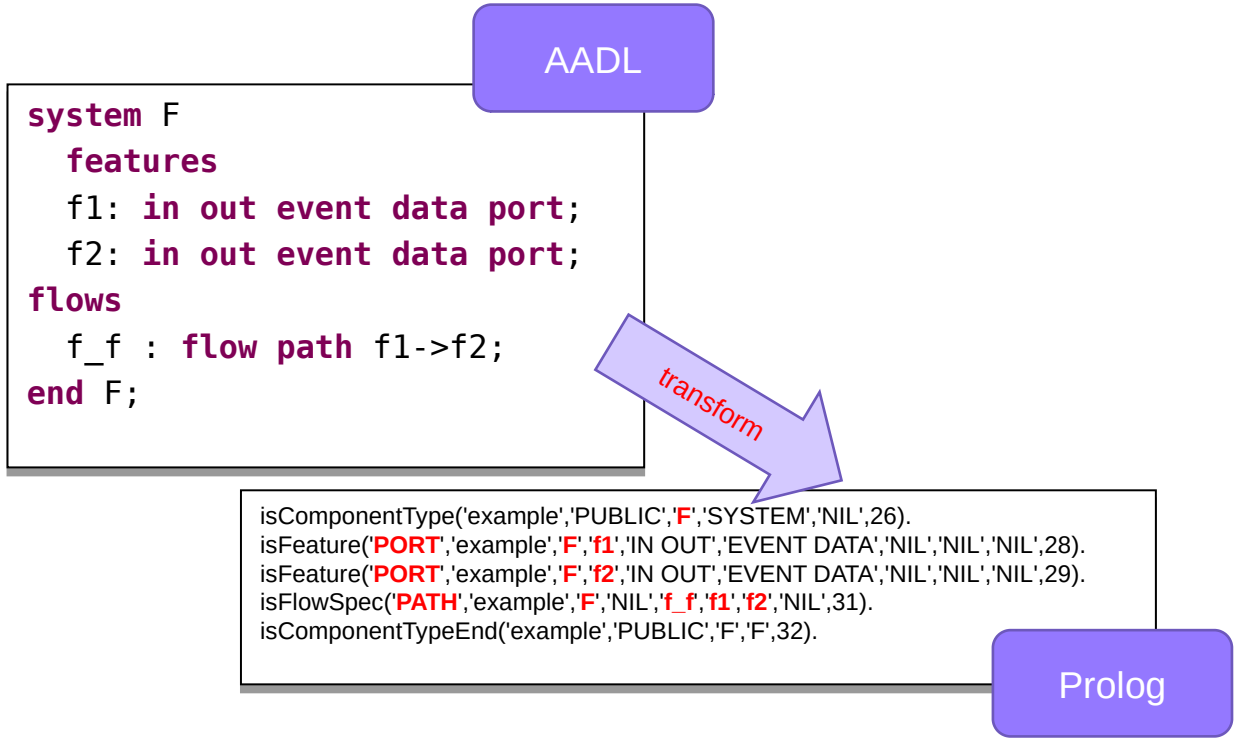
FRENCH INSTITUTES OF TECHNOLOGY

Identifying interference sources

Modeling for interference analysis

- Processeurs are (very) complex
  - TC277, more than 5000 pages
- Documentation is developer-oriented
- Documentation is not always correct / complete

- **How to ensure the completeness of the analysis?**
  - Identify “components”, flows of transactions
  - Capture elements using AADL
  - Translate to Prolog



Identifying interference sources

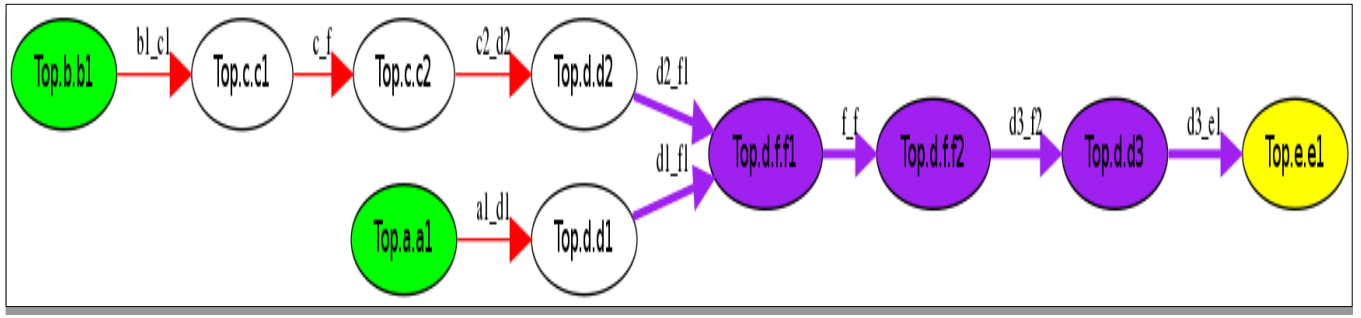
Modeling for interference analysis

- Processeurs are (very) complex
  - TC277, more than 5000 pages
- Documentation is developer-oriented
- Documentation is not always correct / complete

- **How to ensure the completeness of the analysis?**
  - Identify “components”, flows of transactions
  - Capture elements using AADL
  - Translate to Prolog
  - Query Prolog

```
?- interference2(Initiator_1, Target_1, Initiator_2, Target_2,
Crossings).
Initiator_1 = 'Top.a.a1',
Target_1 = Target_2, Target_2 = 'Top.e.e1',
Initiator_2 = 'Top.b.b1',
Crossings = ['Top.d.f.f2', 'Top.d.f.f1', 'Top.d.d3'] ;

Initiator_1 = 'Top.b.b1',
Target_1 = Target_2, Target_2 = 'Top.e.e1',
Initiator_2 = 'Top.a.a1',
Crossings = ['Top.d.f.f2', 'Top.d.f.f1', 'Top.d.d3'].
```



Identifying interference sources

Modeling for interference analysis

- Processeurs are (very) complex
  - TC277, more than 5000 pages
- Documentation is developer-oriented
- Documentation is not always correct / complete

- **How to ensure the completeness of the analysis?**
  - Identify “components”, flows of transactions
  - Capture elements using AADL
  - Translate to Prolog
  - Query Prolog

Identifying interference sources

Modeling for interference analysis

- Processeurs are (very) complex
  - TC277, more than 5000 pages
- Documentation is developer-oriented
- Documentation is not always correct / complete

- **How to ensure the completeness of the analysis?**
  - Identify “components”, flows of transactions
  - Capture elements using AADL
  - Translate to Prolog
  - Query Prolog

How to improve the docs? (standard format?)

Provide micro-benchmarks?

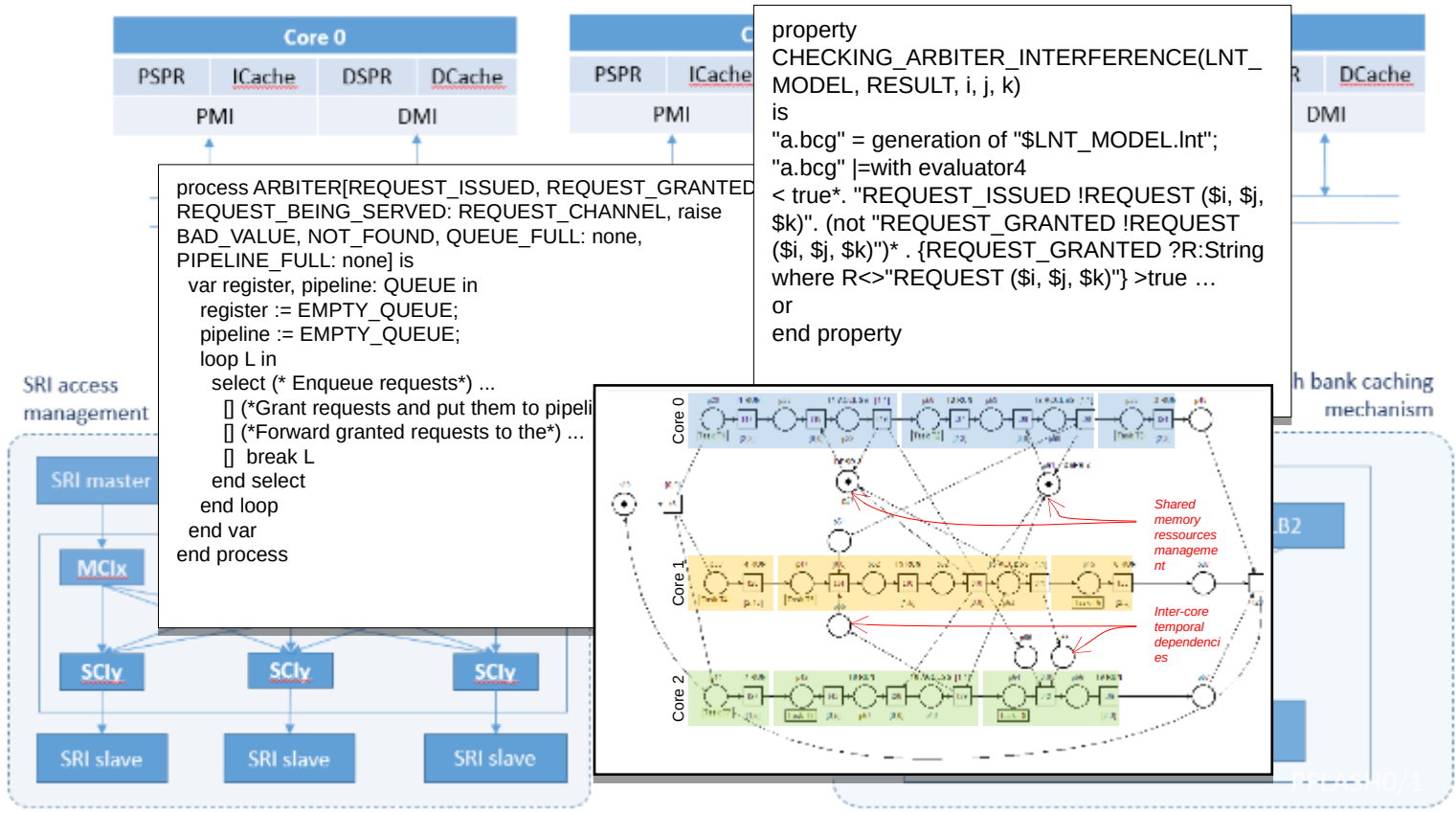
Use AI techniques?

How to improve accuracy?

Identifying interference sources

Using model checking

- Formal modeling of architecture using LNT and Fiacre
- Two methods
  - Patcheck
  - SynCheck



Identifying interference sources

Using model checking

- Formal modeling of architecture using LNT and Fiacre
- Two methods
  - **Patcheck**
  - SynCheck

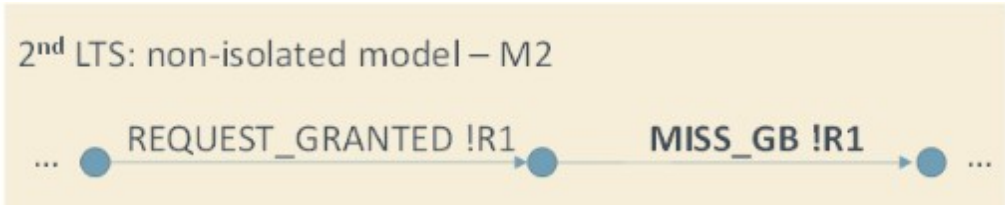
Without interference

With interference

Identifying interference sources

Using model checking

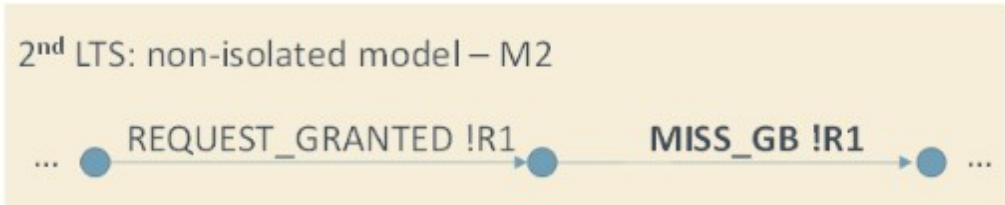
- Formal modeling of architecture using LNT and Fiacre
- Two methods
  - Patcheck
  - **SyncCheck**



Identifying interference sources

Using model checking

- Formal modeling of architecture using LNT and Fiacre
- Two methods
  - Patcheck
  - **SyncCheck**



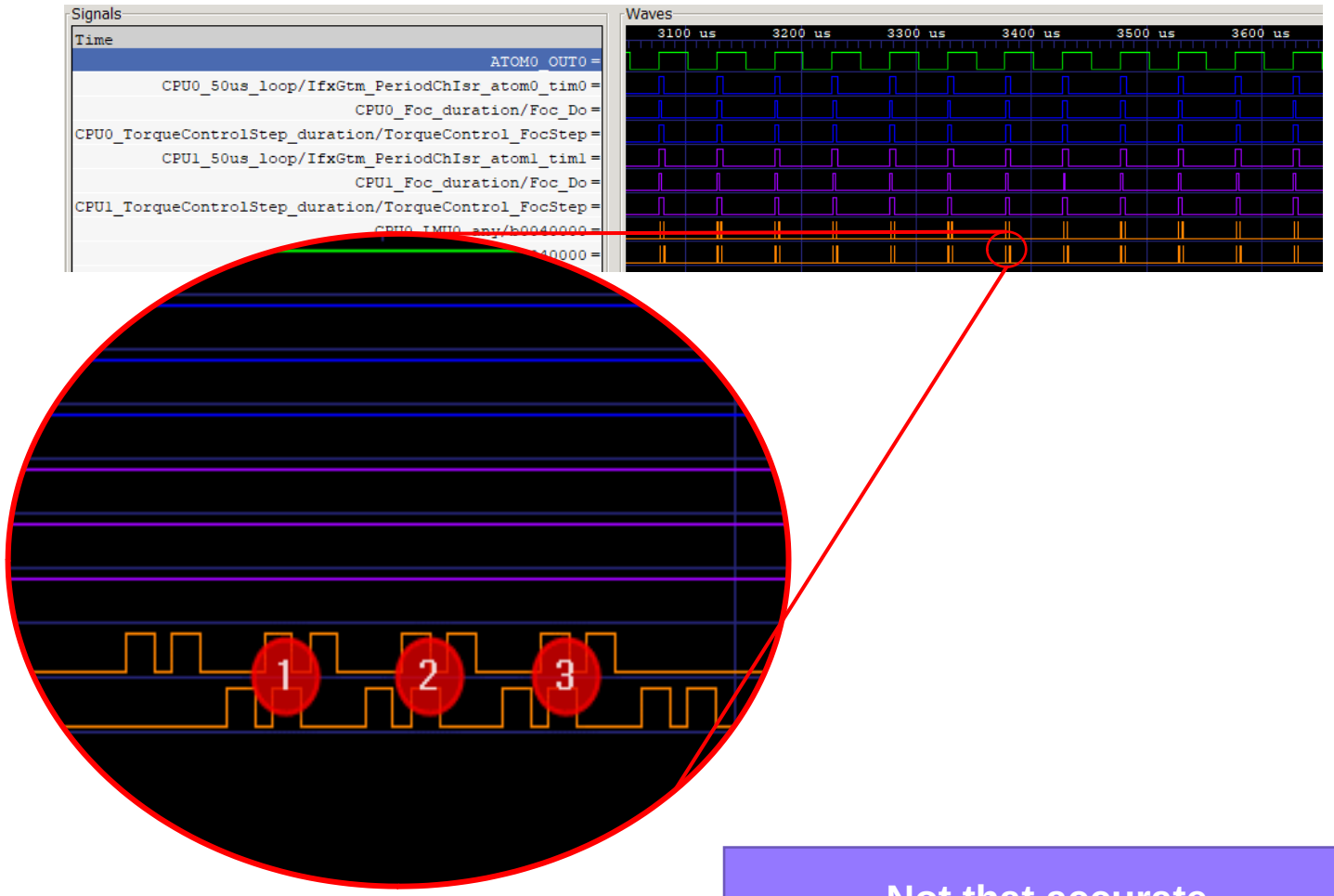
How to build the model?

How to trust the model?

Identify interferences

Using simulation

- Using the VLAB virtual platform
  - Timing modeling is not extremely accurate (this is not a cycle accurate simulator)
  - Identification of multiple transactions in the same time interval.



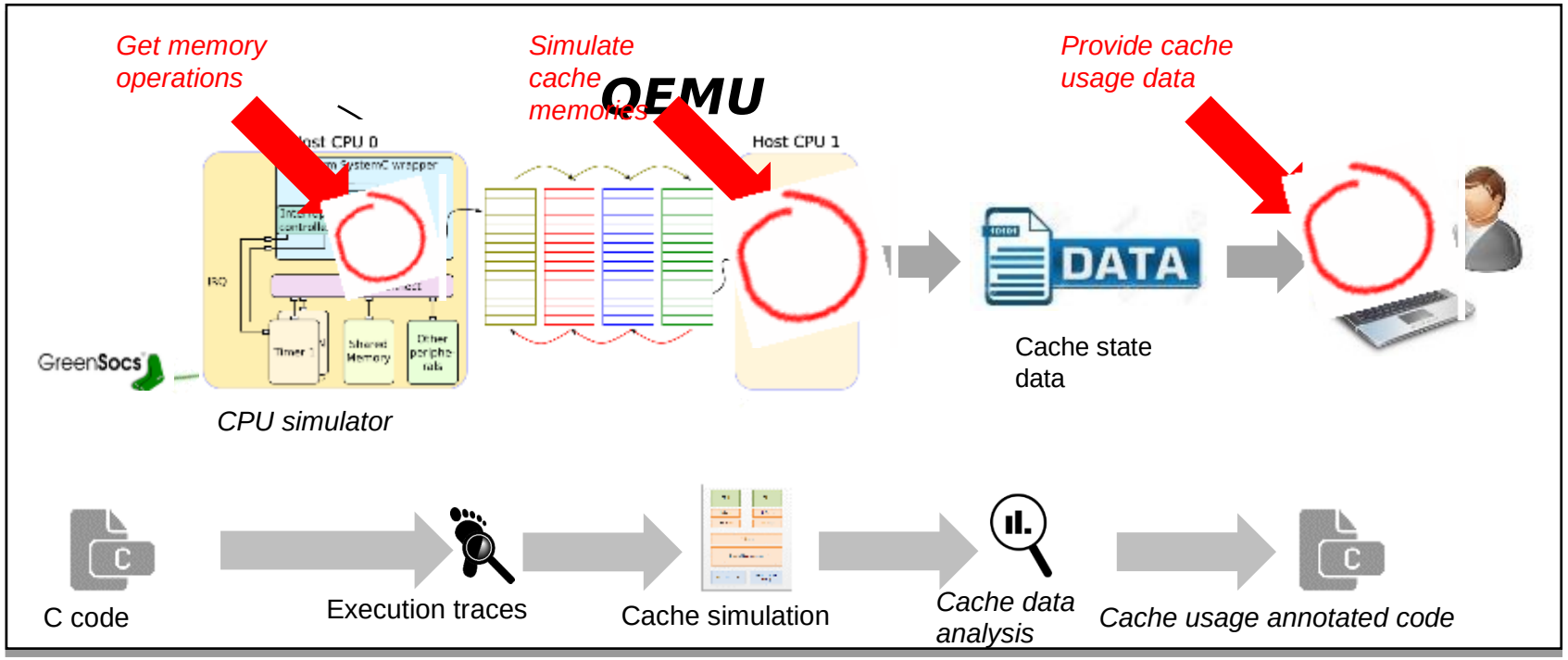
Not that accurate

Does not really identify interferences

Reduce interferences

Identify time-sensitive code

- Example: early empirical cache-sensitivity analys
- QEMU model
- First, provide cache usage data



Reduce interferences

Identify timle-sensitive code

- Example: early empirical cache-sensitivity analys
- QEMU model
- First, provide cache usage data
- Application
  - Xilinx ZCU102 Board, with a single core cortex-r5
  - Running Polybench collection

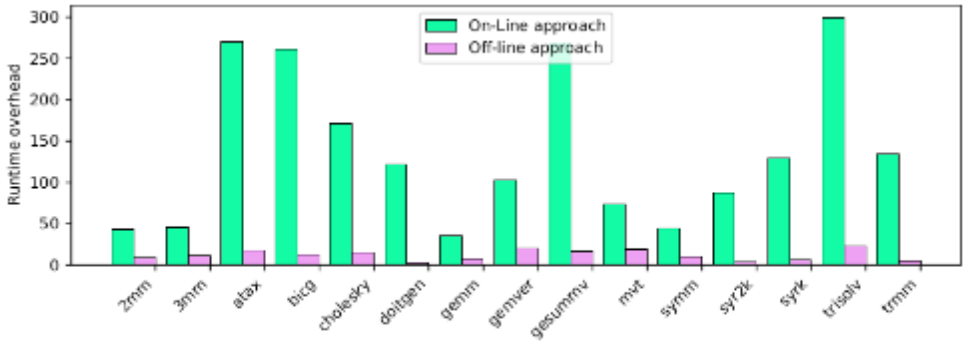


Fig. 6: Runtime overhead compared to vanilla QEMU

Average Overhead: 12%

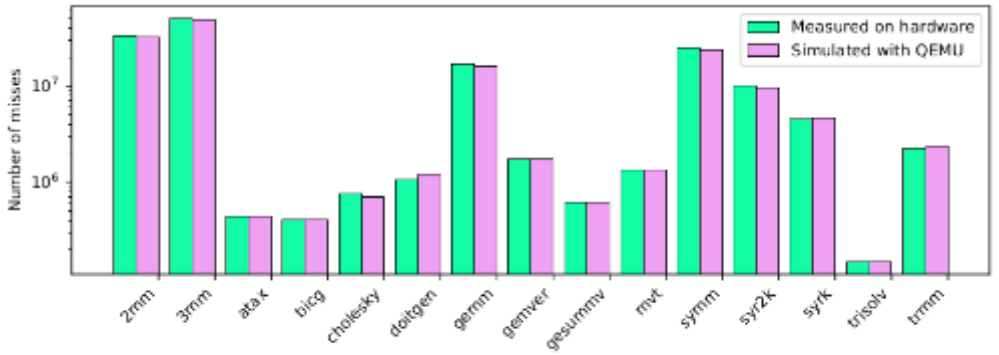


Fig. 5: Total number of misses

Average Relative Error: 3.27%

Reduce interferences

Identify timle-sensitive code

- Example: early empirical cache-sensitivity analys
- QEMU model
- First, provide cache usage data
- Application
  - Xilinx ZCU102 Board, with a single core cortex-r5
  - Running Polybench collection
- Second, provide user-level data

```
64  void consumer0::thread() {
65
66      unsigned int array[SIZE][SIZE], copy[SIZE][SIZE] = {0};
67
68
69
70      while(1) {
71
72          ReadData();
73
74          for (unsigned int i = 0 ; i < SIZE ; i++) {
75
76              for (unsigned int j = 0 ; j < SIZE ; j++) {
77
78                  copy[i][j] = array[i][j];
79
80              }
81
82          }
83
84          SendResult();
85
86      }
87
```

```
copy[i][j] = array[i][j];
```

Lines [76;80] -> **88% hits**, 12% misses
Lines [74;82] -> **86% hits**, 14% misses
Lines [70;86] -> 50% hits, 50% misses

```
copy[j][i] = array[j][i];
```

Lines [76;80] -> **98% hits**, 2% misses
Lines [74;82] -> **100% hits**, 0% misses
Lines [70;86] -> 50% hits, 50% misses

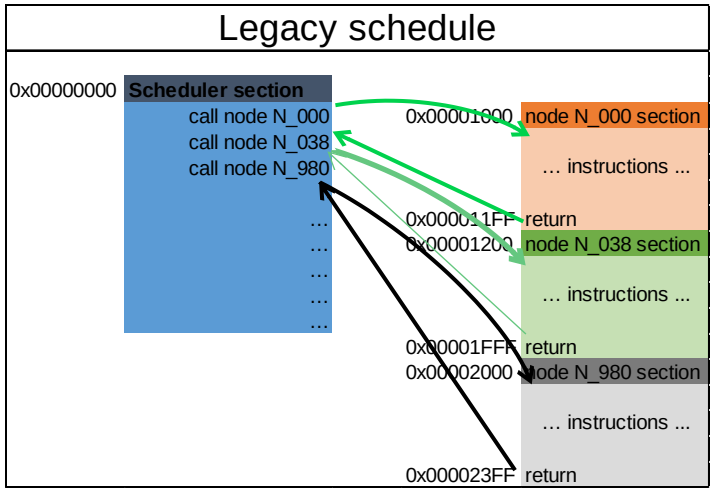
PoC limited to caches

Provide user-level data about interferences

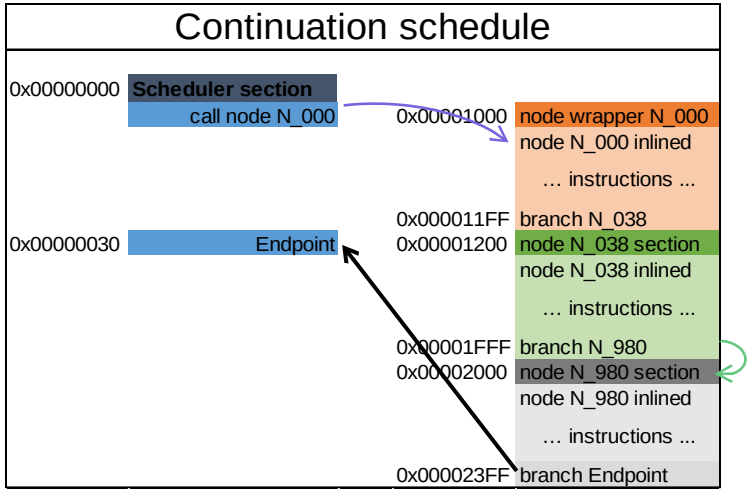
Reduce interferences

On synchronous code

- **On Lustre code**
  - Application to INRIA's LOPHT tool (KAIROS)
  - Reduction of DDR page changes



Each call induce
- Instructions pipeline break (at core level)
- usually a DDR page change

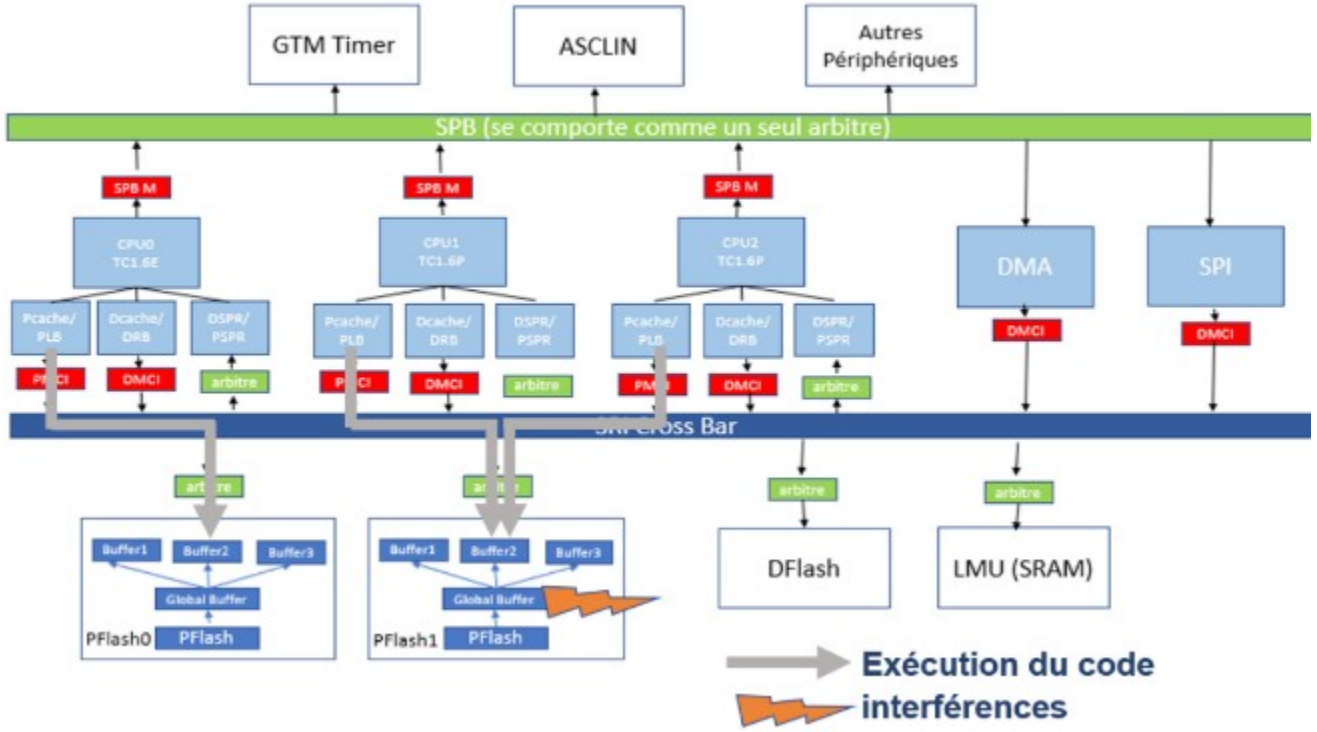


- Linear code instructions
- DDR page change only when code size oversize a page
- Require a timed-triggered OS

Reduce interferences

On synchronous code

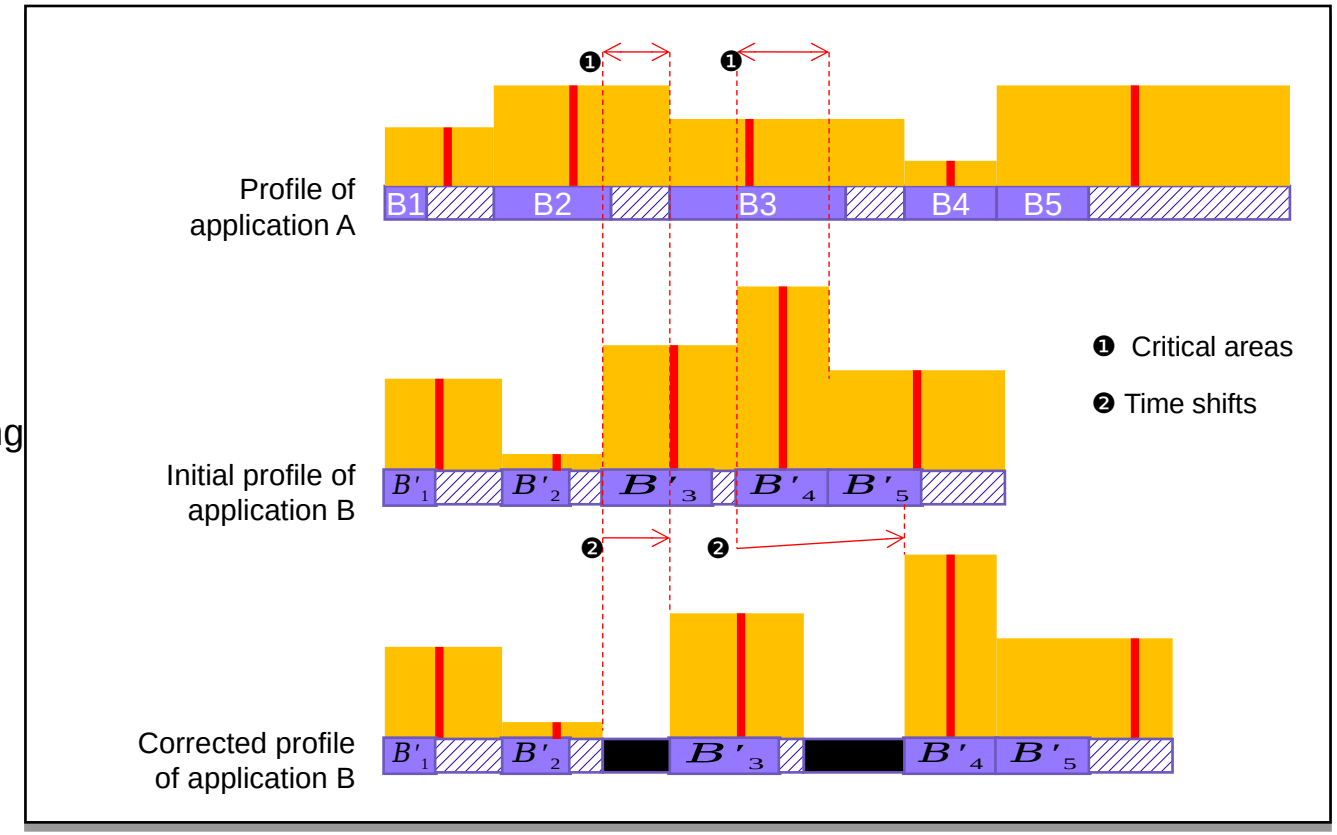
- **On Lustre code**
  - Application to INRIA's LOPHT tool (KAIROS)
  - Reduction of DDR page changes
  - Reduction of flash-level interferences using prefetching (on-going work)



Reduce interferences

On synchronous code

- **On Lustre code**
  - Application to INRIA's LOPHT tool (KAIROS)
  - Reduction of DDR page changes
  - Reduction of flash-level interferences using prefetching (on-going work)
- **On PsyC code** (ASTERIOS)
  - Prevent simultaneous accesses to resources



Static WCET analysis

Building the model

- A tedious activity…

	Instructions vs cycle			0	1	2	3	4	5	6	7	8	9
1	mtcr 0xfc00,d15	FE	DE	**EX1**	EX2								
2	movh.a a2,0x7000		FE	DE	EX1	EX2							
3	st.w [a2] 0, d8			FE	DE	EX1	EX2						
4	st.w [a2] 16, d8				FE	DE	EX1	EX2					
5	st.w [a2] 32, d8					FE	DE	EX1	EX2				
6	ld.w d8,[a2] 64						FE	DE	EX1	EX2			
7	add d7, d8, d5							FE	DE		EX1	EX2	
8	mov d0,0								FE		DE	EX1	EX2
9	mtcr 0xfc00,d0										FE	DE	**EX1**

SB0						7000	7016	7032					
SB1							7000	7016	7032				
SB2								7000	7016	7032			
SB3									7000	7016	7032		
Memory Access									r7064	w7000	w7016	w7032	

Better documentation?

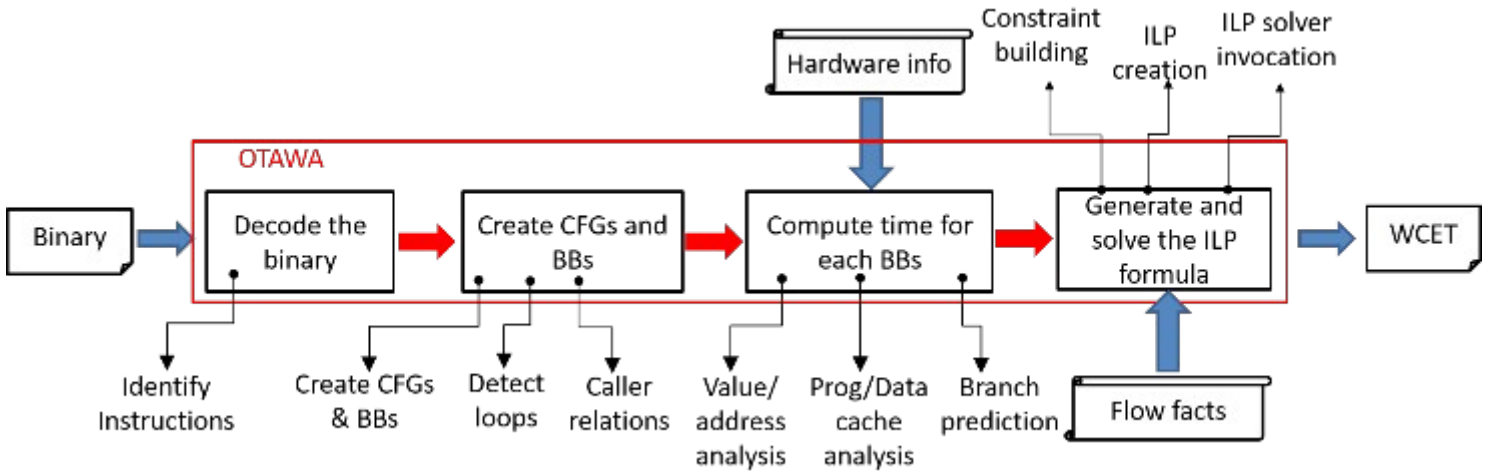
Documentation for timing analysis?

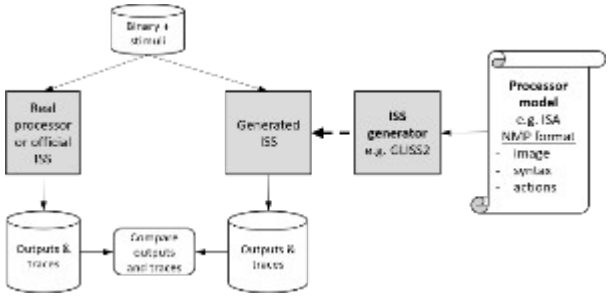
FRENCH INSTITUTES OF TECHNOLOGY

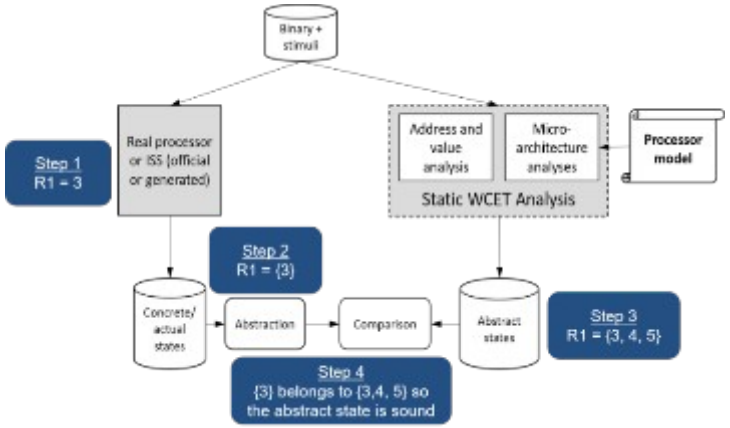
Static WCET analysis

How to gain confidence on the tool

- Two levels verification
  - Verifying the ISS
    - ISS generated from NMP files used to decode instructions in OTAWA
    - First step to support a new architecture
    - Compare the “processor state” between TSIM and OTAWA-ISS
      - Register values
      - Memory accesses
  - Verifying the abstract model
    - Is the value/address analysis correctly implemented?
    - Are “semantic instructions” correctly implemented for TriCore



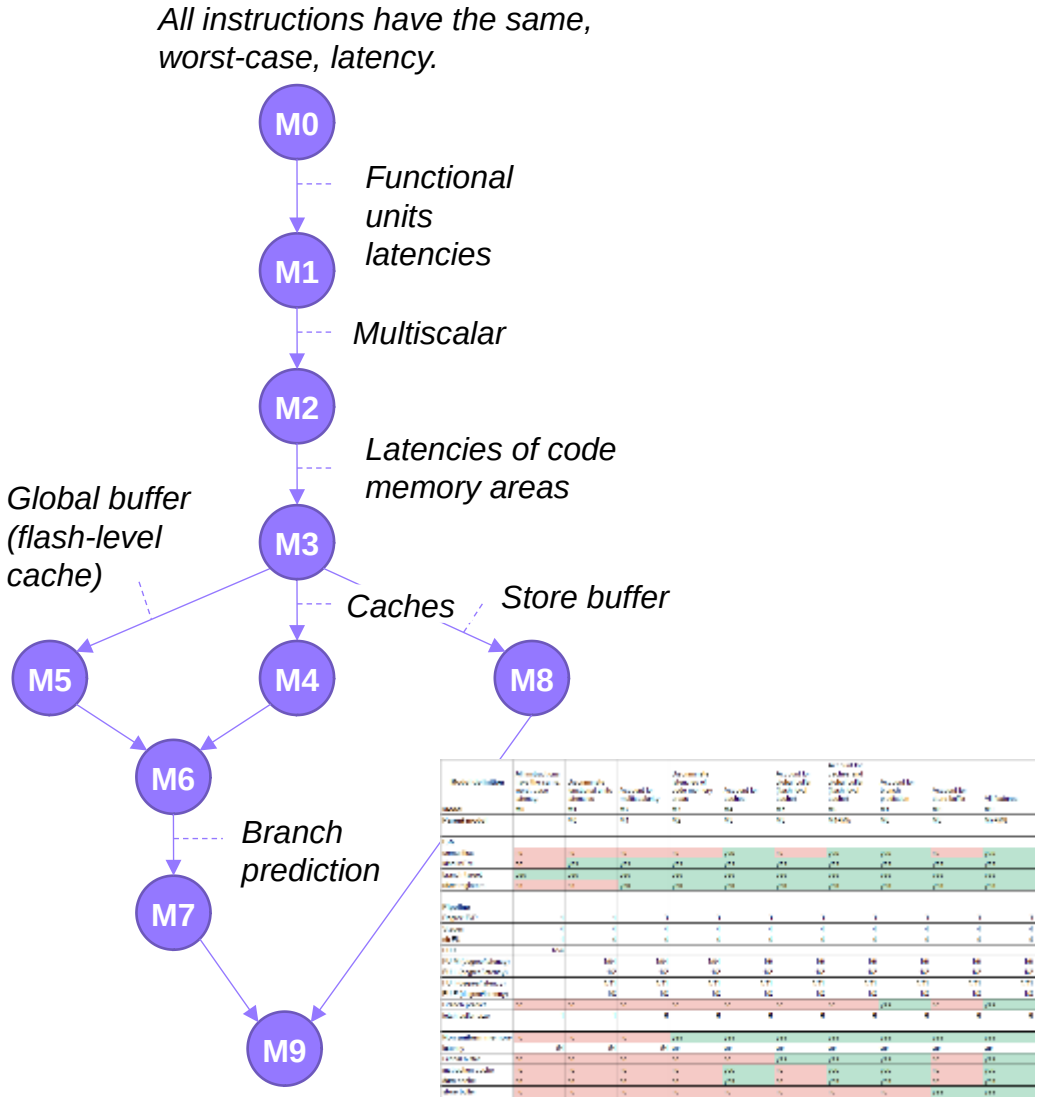




Reduce cost of analysis

How to trade-off accuracy and cost

- WCET analyzers based on AI are costly to develop
- Can we focus the effort on the most "pertinent" parts of the component (the one with the highest contribution)?



On-going work…

Predictability by construction

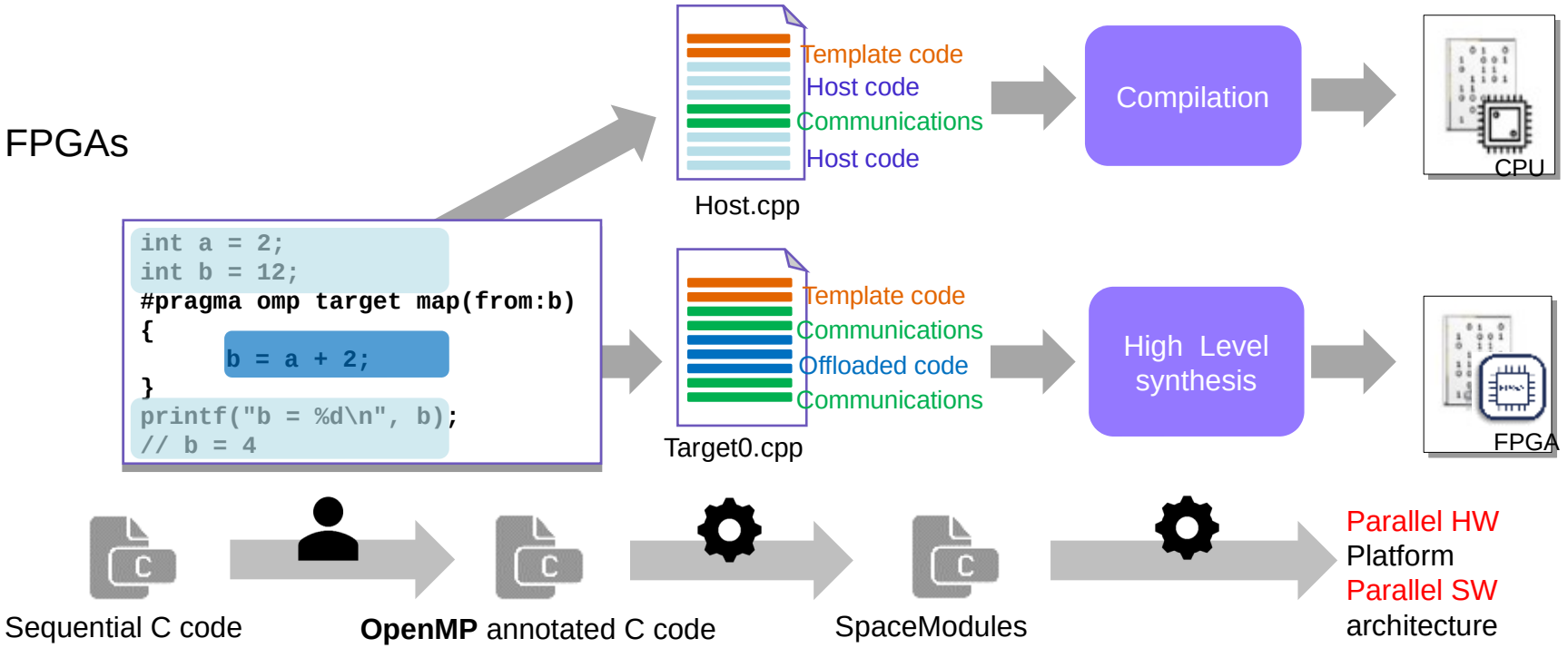
How to build a predictable platform?

Deploy functions to hardware

Deploy to FPGA

- Leverage SoCs embedding FPGAs (Zynq, Dahlia, etc.)
- Exploit parallelism
- By product of performance enhancement

- Example: OpenMP to HW

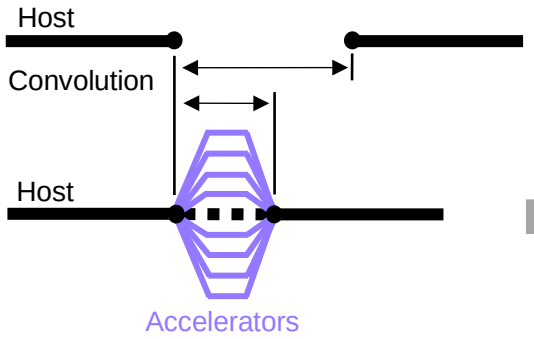


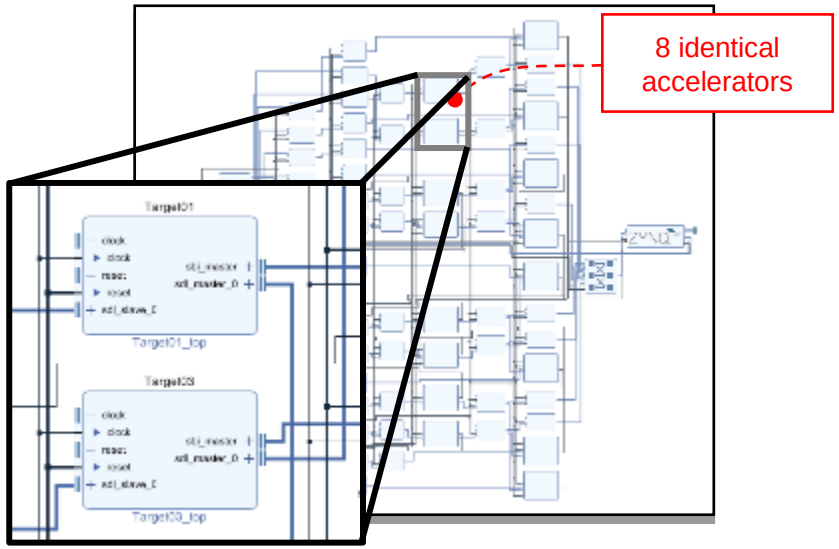
Deploy functions to hardware

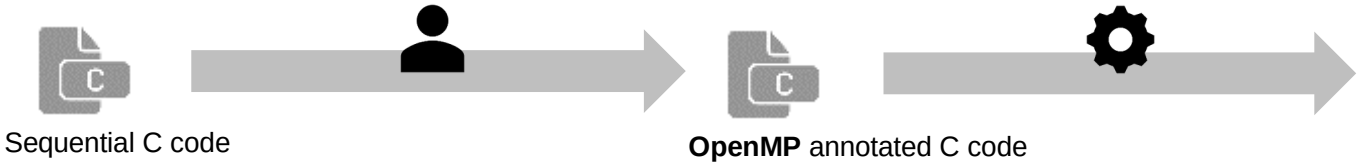
Deploy to FPGA

- Leverage SoCs embedding FPGAs (Zynq, Dahlia, etc.)
- Exploit parallelism
- By product of performance enhancement

- Example: OpenMP to HW







Needs a FPGA…

… usually dedicated to other demanding computations…

Deploy functions to hardware

New issues with parallelism

- Parallelism raises "new" issues with respect to determinism
  - Deadlocks
  - Race conditions…
- Parallelization frameworks generally not designecd for safety critical systems…

Provide a “deterministic” version OpenMP

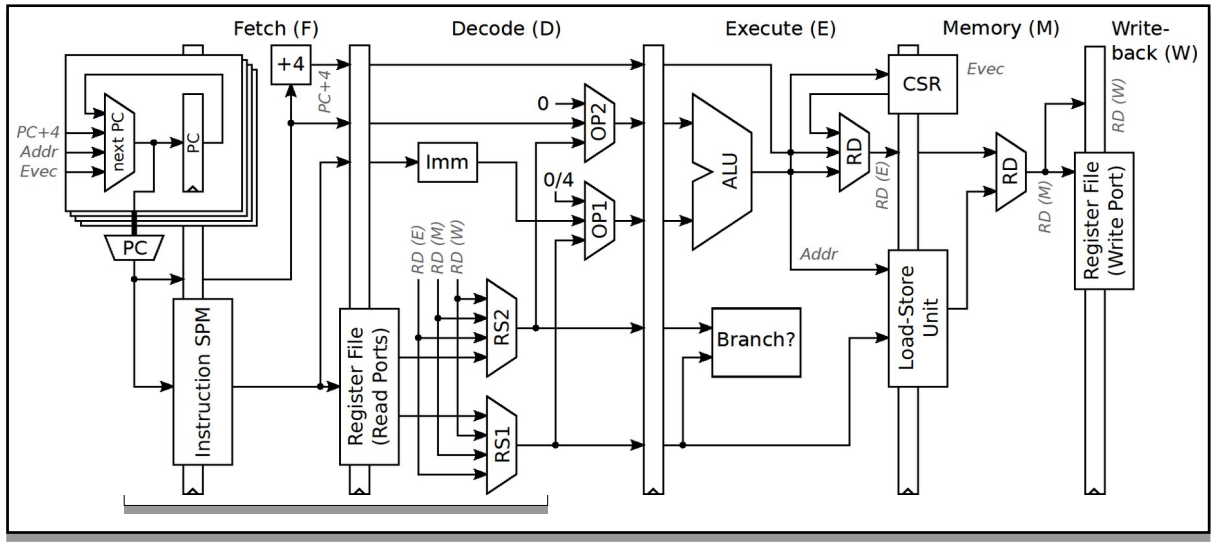
Prevent / detect non-deterministic effects

Determinism by contruction

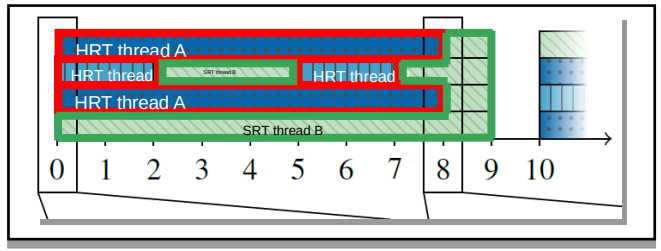
Use the PL to implement a timing-analysis friendly processor

- Split the problem (not every processing have the same level of temporal requirements…)
- Deploy time-critical function on deterministic processors
- Example: Berkeley's FlexPRET

- Combination of FlexPRET anb synchronous execution model (Lustre/LOPHT and ForeC)



Berkeley's FlexPRET (M. Zimmer's PhD thesis)



Pipeline shared by hard and soft and HW threads

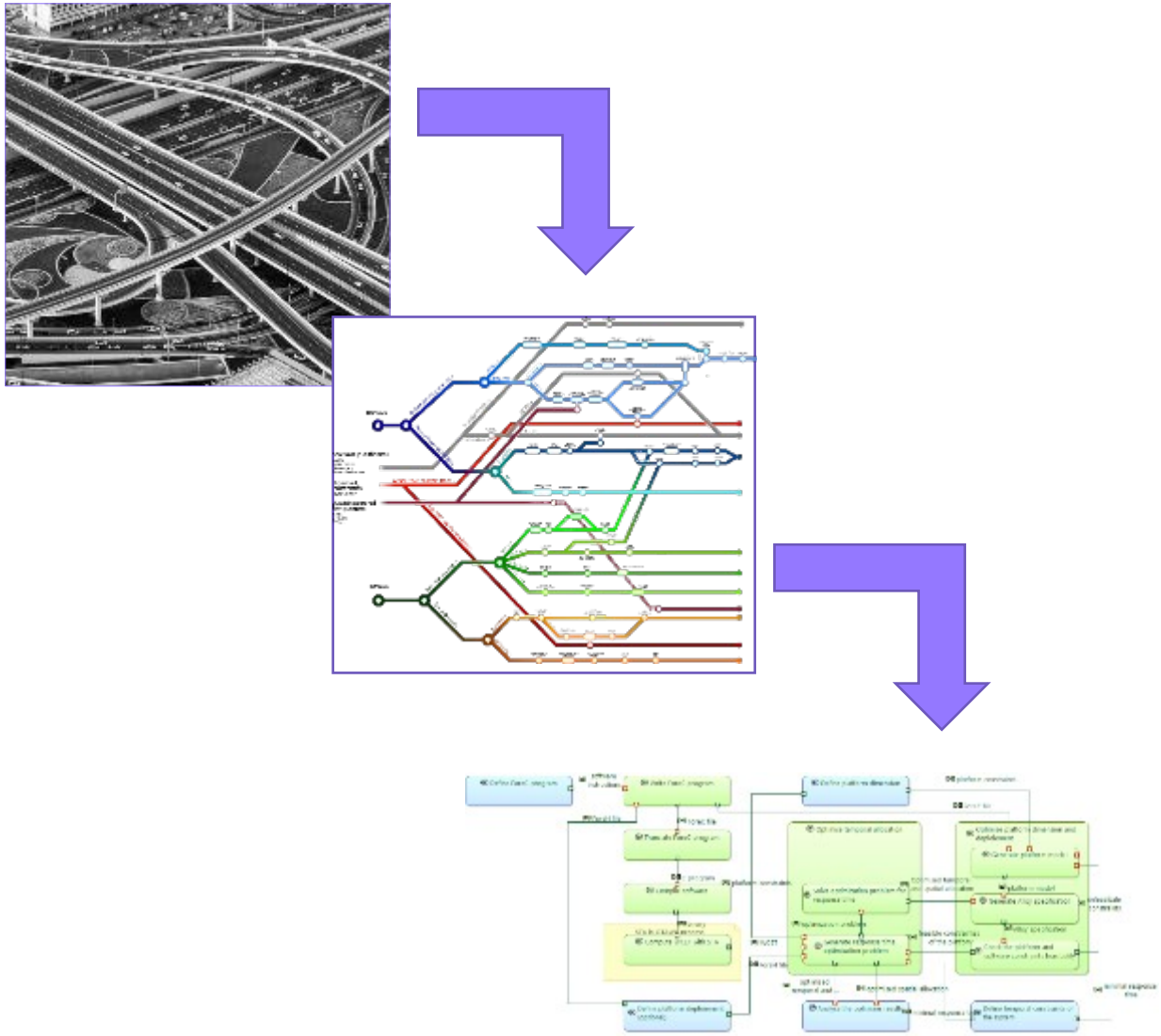
Performances?

"Exotic" architecture…

FRENCH INSTITUTES OF TECHNOLOGY

Conclusion

Things are getting clearer…



- But problems are getting more (and more) complicated…
- But there is still quite a lot of work to be done (see all the boxes of the presentation)
- Guidance is still missing for industrial partners: *Which approach shall I use to use for what result and with what confidence on the result?…*

References

Publications

- Interference analysis
  - W.-T. Sun, E. Jenn, H. Cassé, and T. Carle, ‘Automatic Identification of Timing Interferences on Multi-Core Processor in a Model-Based Approach’, presented at the COMPAS 2019, Anglet, 28/6 2019.
  - V. A. Nguyen, E. Jenn, W. Serwe, F. Lang, and R. Mateescu, ‘Using Model Checking to Identify Timing Interferences on Multicore Processors’, in *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, Toulouse, France, Available at https://hal.inria.fr/hal-02462085
  - A. Ferlin, E. Jenn, and M. Kaufmann, ‘Accounting for interferences in the design of Time-Triggered Applications’, in *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, Toulouse, France, Jan. 2020, Accessed: Apr. 28, 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02469116.
  - F. Guet, L. Santinelli, J. Morio, G. Phavorin, and E. Jenn, ‘Toward Contention Analysis for Parallel Executing Real-Time Tasks’, 18th International Workshop on Worst-Case Execution Time Analysis (WCET 2018), Barcelona, Spain, Jul. 2018
- WCET analysis
  - W.-T. Sun, E. Jenn, and H. Cassé, ‘Validating static WCET analysis: a method and its application’, presented at the 19th International Workshop on Worst-Case Execution Time Analysis, Stuttgart, Germany, Jul. 2019., Available at https://drops.dagstuhl.de/opus/volltexte/2019/10771/.
  - W.-T. Sun, E. Jenn, and H. Cassé, ‘Build Your Own Static WCET analyser: the Case of the Automotive Processor AURIX TC275’, in *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, Toulouse, France, Available at https://hal.archives-ouvertes.fr/hal-02507130

Publications

Parallel applications

- V.-A. Nguyen, W. Serwe, R. Mateescu, and E. Jenn, ‘Hunting Superfluous Locks with Model Checking’, in From Software Engineering to Formal Methods and Tools, and Back, vol. 11865, Springer Verlag, 2019, p. 416-432. Avalable at https://hal.inria.fr/hal-02314088
- R. Leconte, E. Jenn, G. Bois, and H. Guérard, ‘Make Life Easier for Embedded Software Engineers Facing Complex Hardware Architectures’ in *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, Toulouse, France, Available at https://hal.archives-ouvertes.fr/hal-02474476

Deterministic multicores

- N. Hili, A. Girault, and E. Jenn, ‘Worst-Case Reaction Time Optimization on Deterministic Multi-Core Architectures with Synchronous Languages’, presented at the 25th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Hangzhou, China, Aug. 2019.
- A. Girault, N. Hili, E. Jenn, and E. Yip, ‘A Multi-Rate Extension of the ForeC Precision Timed Programming Language for Multi-Cores’, presented at the Forum Design Language, Southampton, United Kigdom

Simulation

- C. Deschamps, M. Burton, E. Jenn, and F. Pétrot, ‘Gathering Memory Hierarchy Statistics in QEMU’, DVCON 2020, (virtual conference), Oct. 2020