Habilitation à Diriger des Recherches High-level Models for Embedded Systems

Matthieu Moy

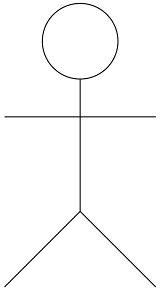
Verimag (Grenoble INP)
Grenoble, France

March 13th 2014

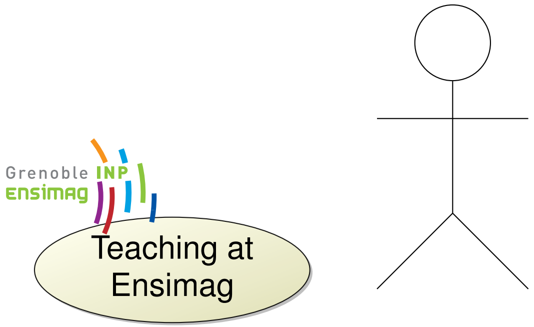
Jury:

Gérard Berry	Professeur au Collège de France	Reviewer
Rolf Drechsler	Professor at TU Bremen, Germany	Reviewer
Marco Roveri	Senior Researcher, Fondazione Bruno Kessler, Italy	Reviewer
Samarjit Chakraborty	Professor at TU Muchen, Germany	Examiner
Benoît Dupont de Dinechin	Directeur Technique, Kalray, France	Examiner
Frédéric Pétrot	Professeur à Grenoble INP, France	Examiner

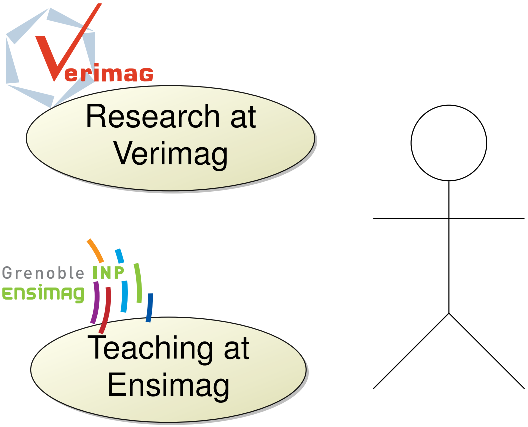
About Me



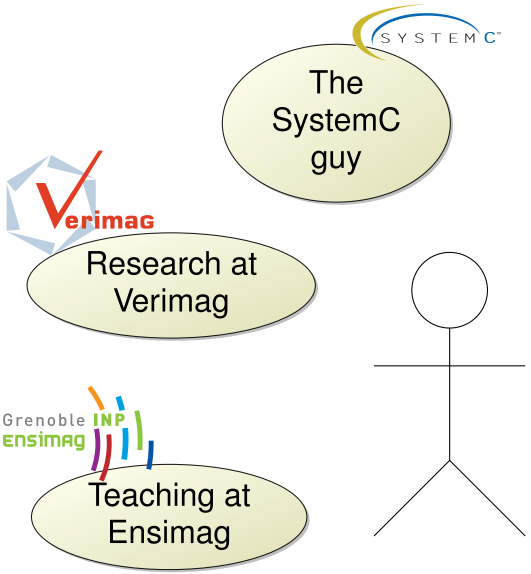
About Me



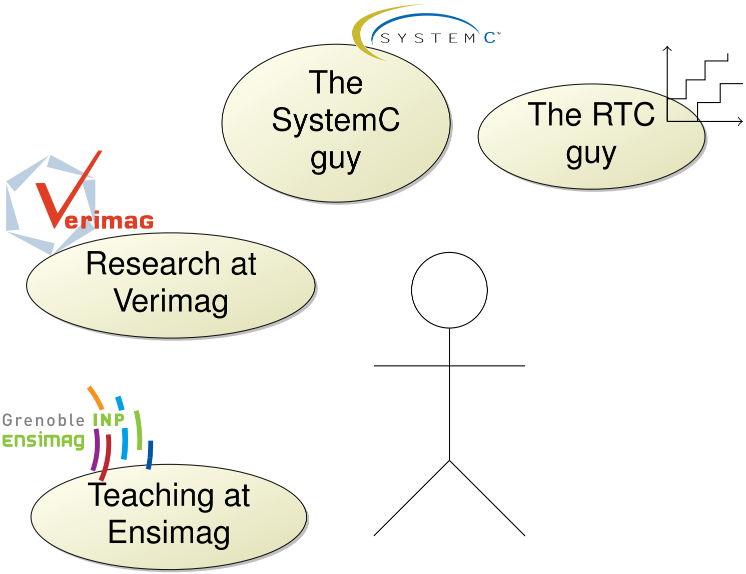
About Me



About Me


SYSTEMC
The SystemC guy
Verimag
Research at Verimag
Grenoble INP Ensimag
Teaching at Ensimag

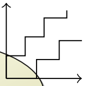
About Me


The SystemC guy
The RTC guy
Research at Verimag
Teaching at Ensimag

About Me

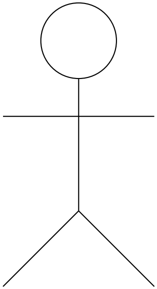
The SystemC guy

The RTC guy



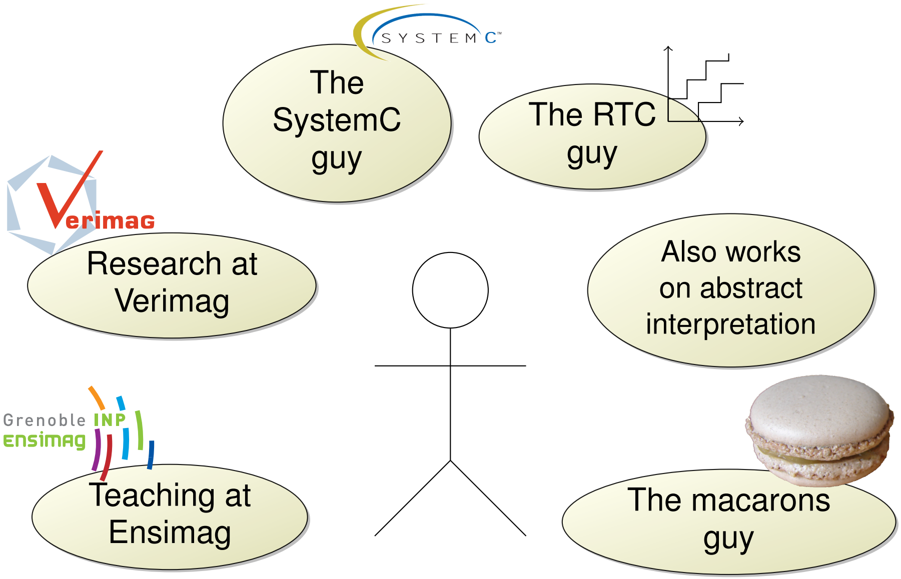
Research at Verimag

Also works on abstract interpretation



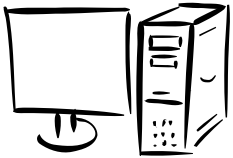
Teaching at Ensimag

About Me


SYSTEMC
The SystemC guy
The RTC guy
Verimag
Research at Verimag
Also works on abstract interpretation
Grenoble INP Ensimag
Teaching at Ensimag
The macarons guy

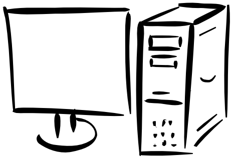
What Are Processors For?

Computers



What Are Processors For?

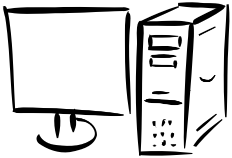
Computers

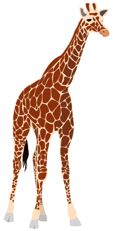


Embedded systems

What Are Processors For?

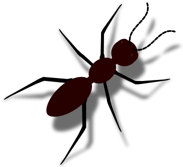
Computers





Vertebrate

Embedded systems

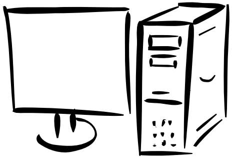


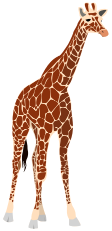
Insects

Source: `http://skepchick.org/2013/03/planet-of-the-arthropods/`

What Are Processors For?

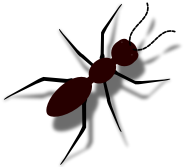
Computers
≈ 2%





Vertebrate
≈ 4%

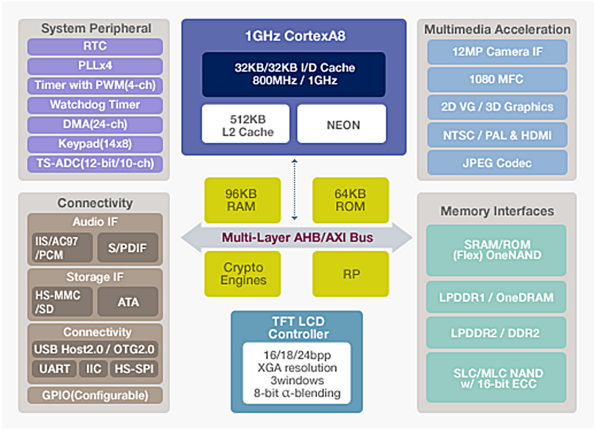
Embedded
systems
≈ 98%



Insects
≈ 96%

Source: `http://skepchick.org/2013/03/planet-of-the-arthropods/`

Prehistory: My Phone (2010)



Source: `http://www.embeddedinsights.com/epd/samsung/samsung-s5pc110.php`

Another Typical Embedded System: my New Camera

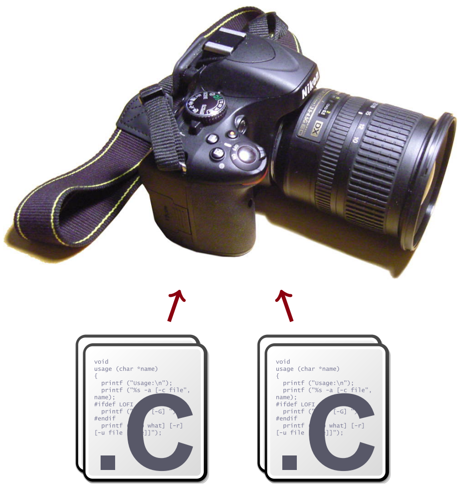
Another Typical Embedded System: my New Camera

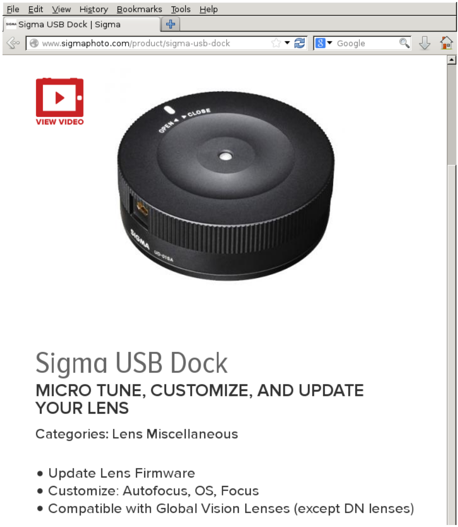
Firmware

Another Typical Embedded System: my New Camera

Firmware A
Firmware B

Another Typical Embedded System: my New Camera




File Edit View History Bookmarks Tools Help
Sigma USB Dock | Sigma
www.sigmaphoto.com/product/sigma-usb-dock
Google
VIEW VIDEO

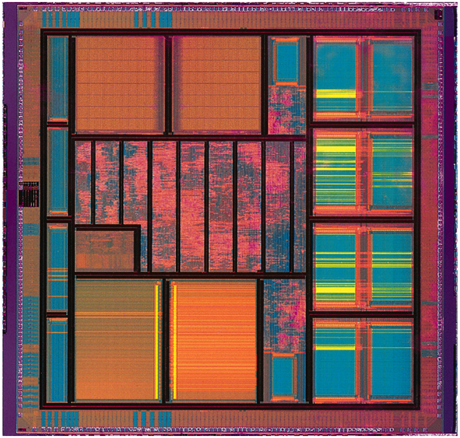
Sigma USB Dock

MICRO TUNE, CUSTOMIZE, AND UPDATE YOUR LENS

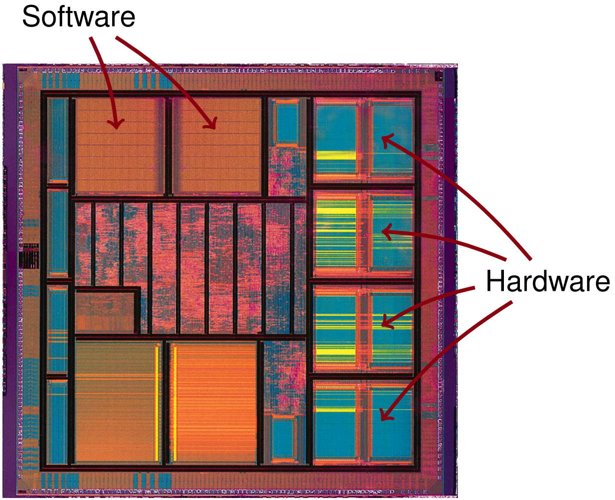
Categories: Lens Miscellaneous

- Update Lens Firmware
- Customize: Autofocus, OS, Focus
- Compatible with Global Vision Lenses (except DN lenses)

Modern Systems-on-a-Chip



Modern Systems-on-a-Chip



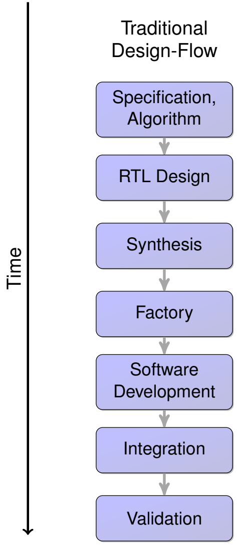
Issue 1: Functional Correctness

Issue 1: Functional Correctness

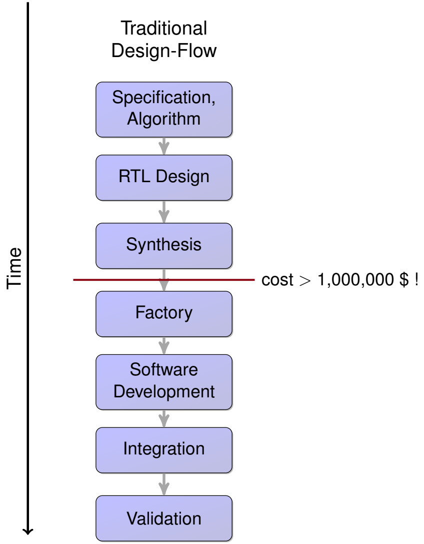
$\downarrow$

Issue 1: Functional Correctness

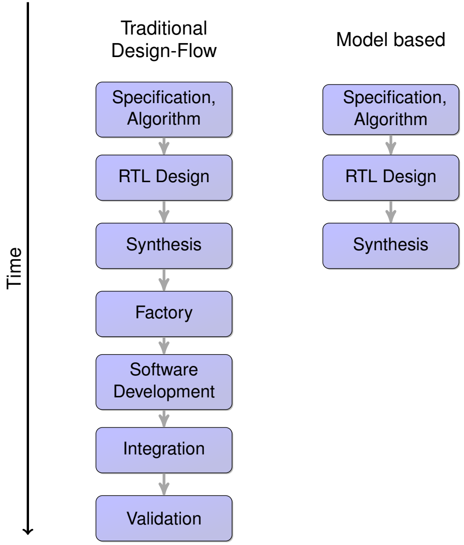
Issue 2: Early Software Development



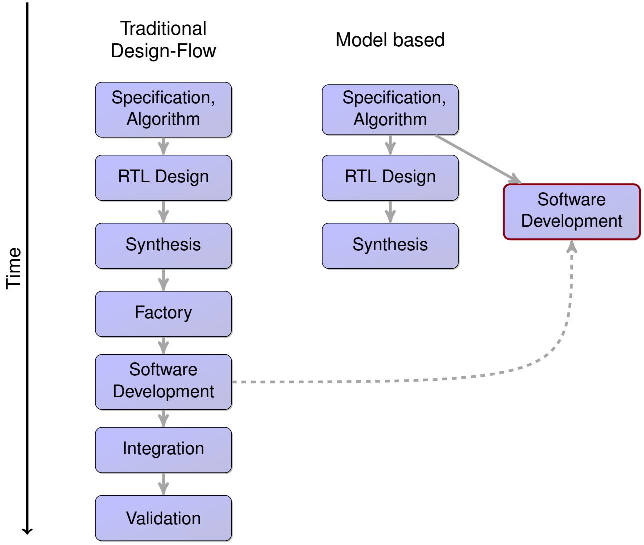
Issue 2: Early Software Development



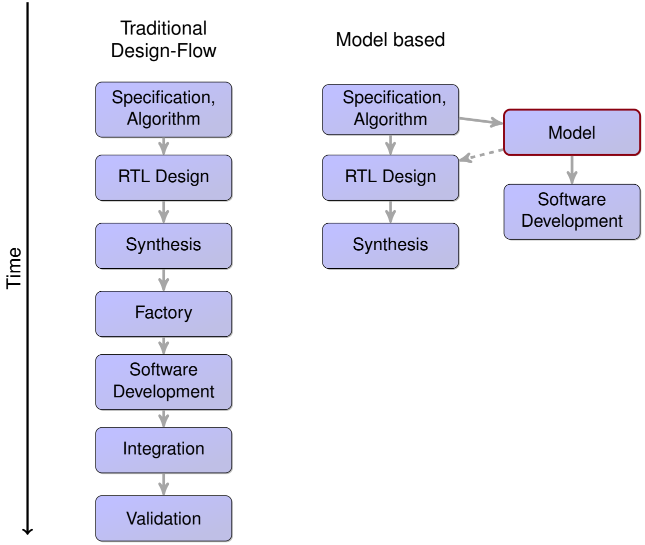
Issue 2: Early Software Development



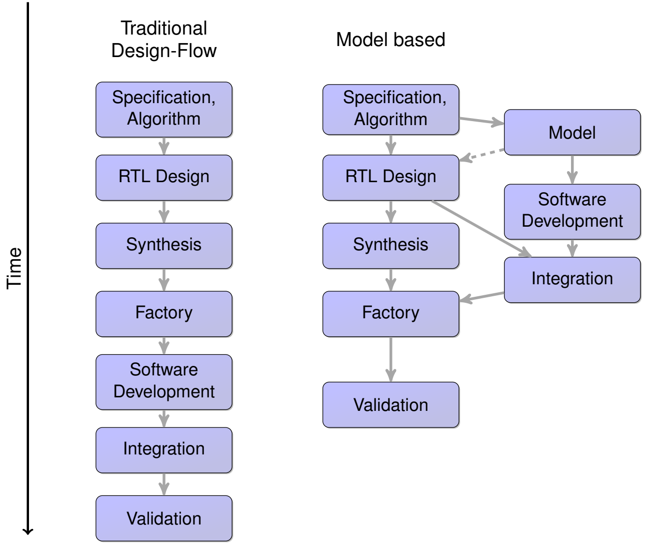
Issue 2: Early Software Development



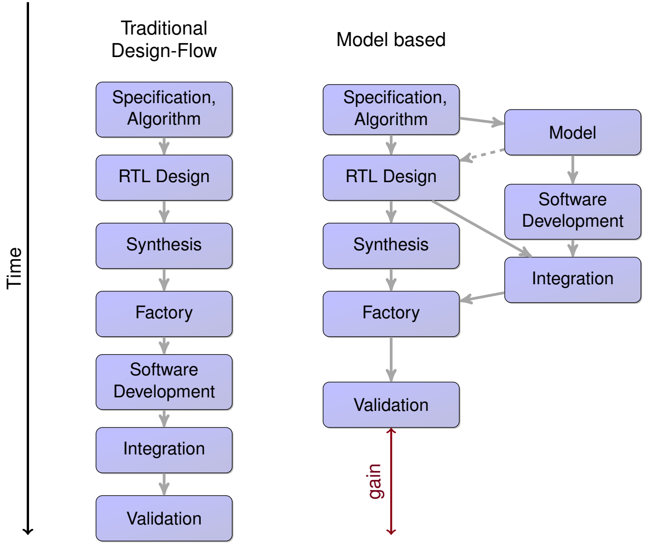
Issue 2: Early Software Development



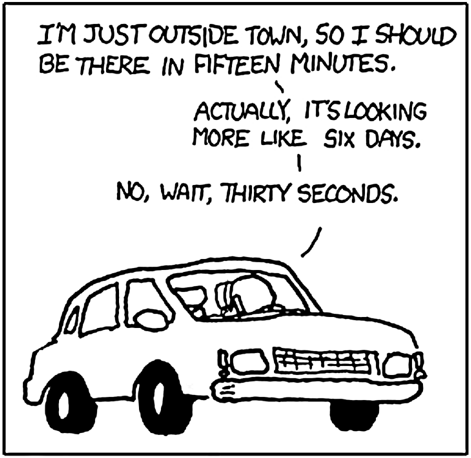
Issue 2: Early Software Development



Issue 2: Early Software Development

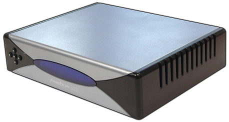


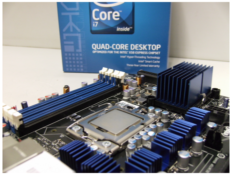
Issue 3: Timing


I'M JUST OUTSIDE TOWN, SO I SHOULD BE THERE IN FIFTEEN MINUTES.
ACTUALLY, IT'S LOOKING MORE LIKE SIX DAYS.
NO, WAIT, THIRTY SECONDS.

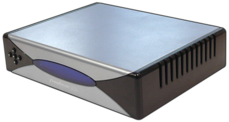
THE AUTHOR OF THE WINDOWS FILE COPY DIALOG VISITS SOME FRIENDS.

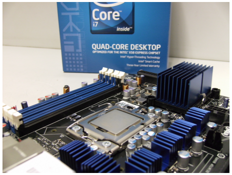
Issue 4: Power and Temperature





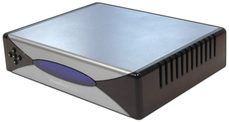
Issue 4: Power and Temperature

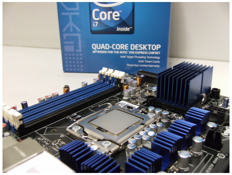




50-130 watt

Issue 4: Power and Temperature

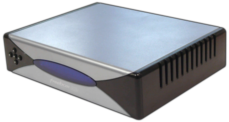




50-130 watt

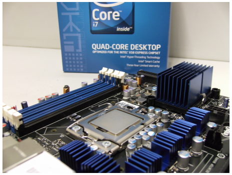
20-30 watt

Issue 4: Power and Temperature



20 watt

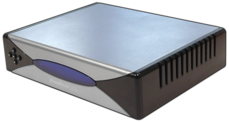
20-30 watt



50-130 watt

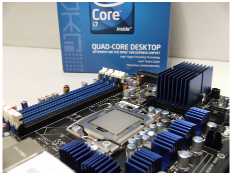
Issue 4: Power and Temperature

$<$ 1 watt



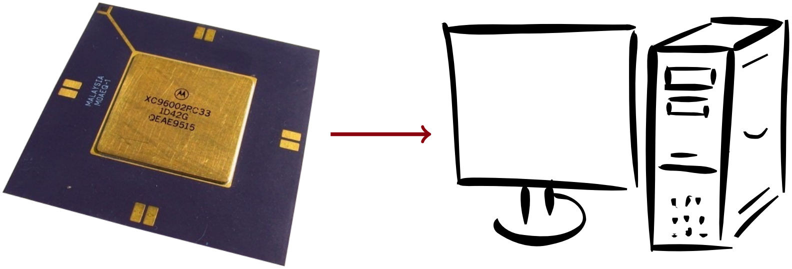
20 watt

20-30 watt

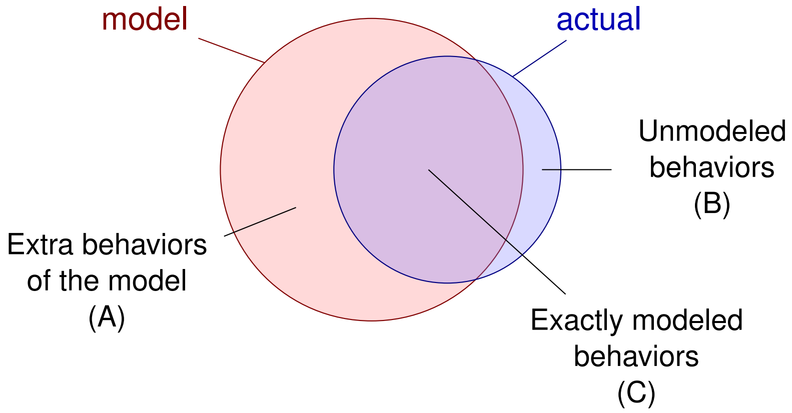


50-130 watt

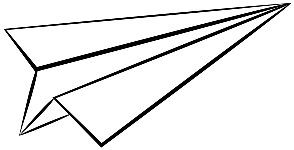
Issue 5: Simulation speed



Issue 6: Model Faithfulness


model
actual
Unmodeled behaviors (B)
Extra behaviors of the model (A)
Exactly modeled behaviors (C)

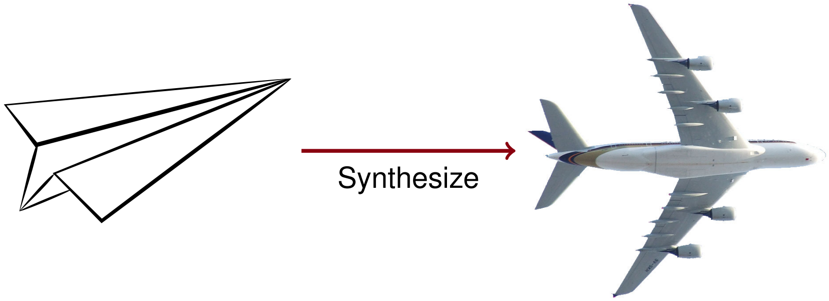
Models and Virtual Prototypes



Model/Prototype

Real system

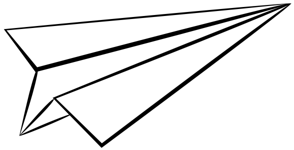
Models and Virtual Prototypes



Model/Prototype

Real system

Models and Virtual Prototypes



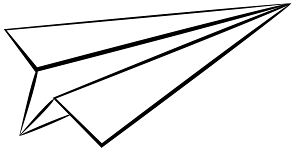
$\xrightarrow{}$

(Synthesize)

Model/Prototype

Real system

Models and Virtual Prototypes

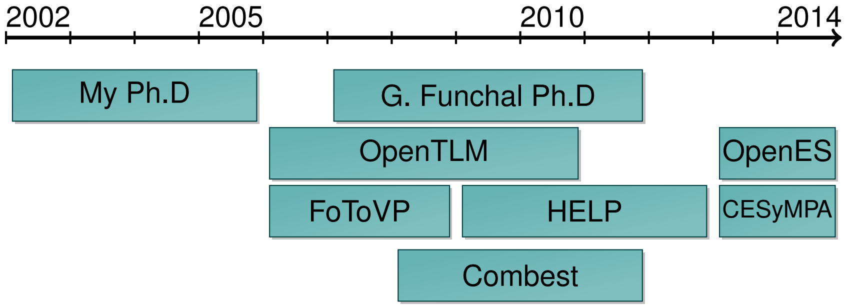


(Synthesize)
Compare
Use as specification
...

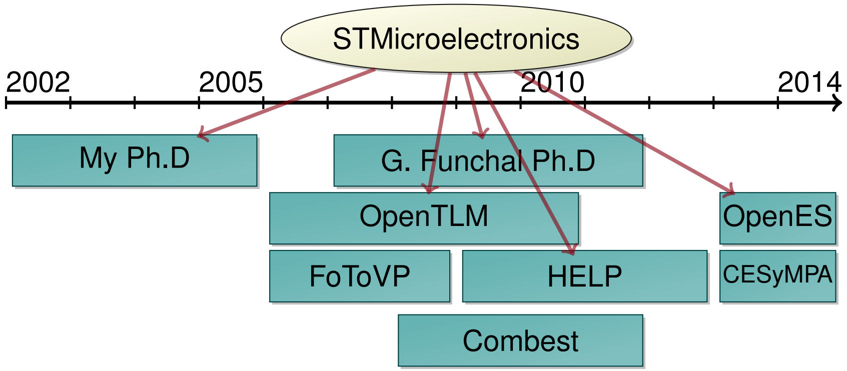
Model/Prototype

Real system

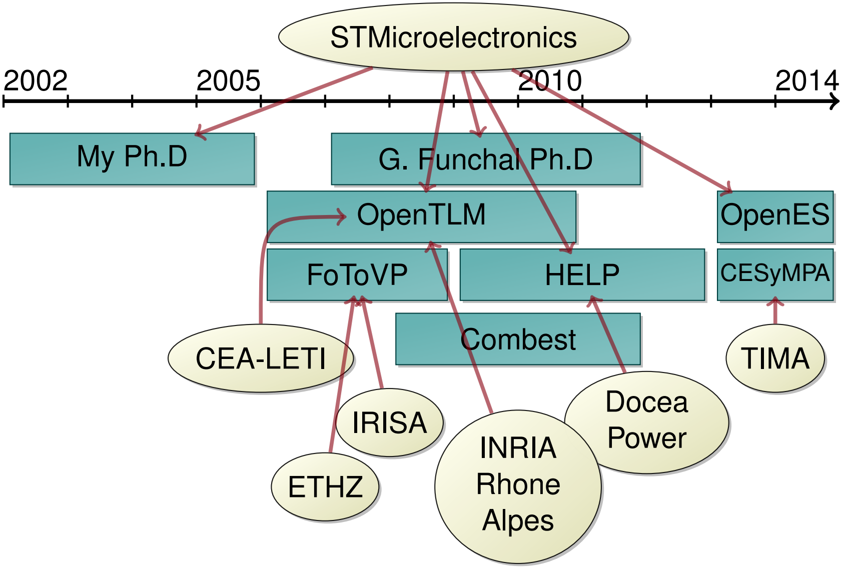
Context and Collaborations of my Work



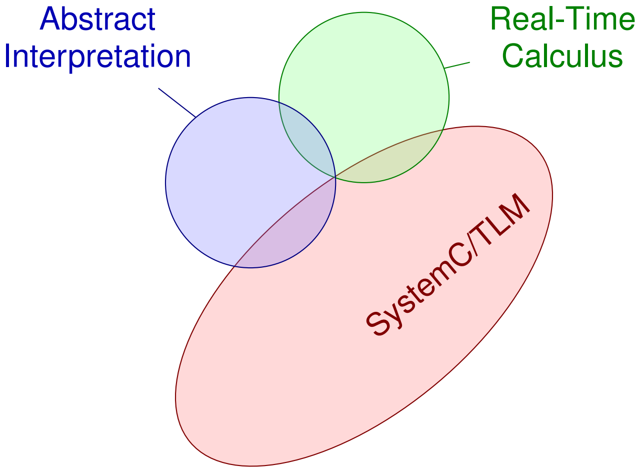
Context and Collaborations of my Work



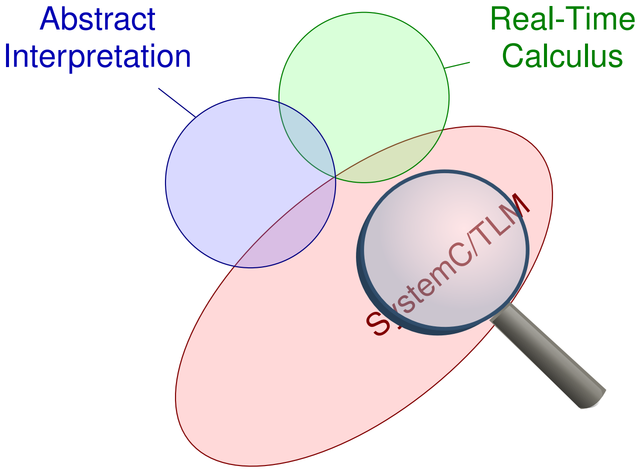
Context and Collaborations of my Work



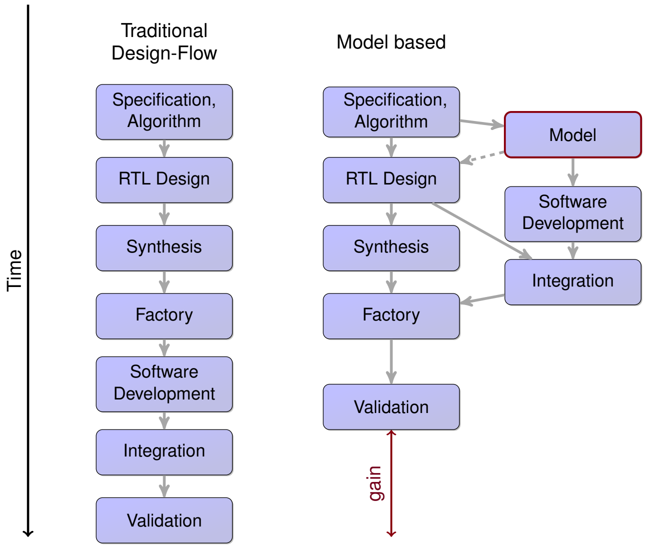
Outline



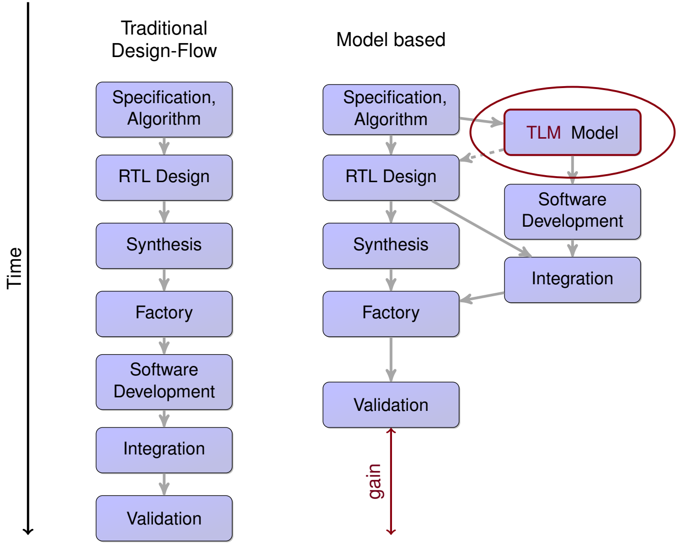
Outline



TLM and Software Development



TLM and Software Development



The Transaction Level Model (TLM): Principles and Objectives

A high level of abstraction,
that appears early in the design-flow

The Transaction Level Model (TLM): Principles and Objectives

A high level of abstraction,
that appears early in the design-flow

$\approx$ 1000 Faster than
low-level simulations (RTL)

The Transaction Level Model (TLM): Principles and Objectives

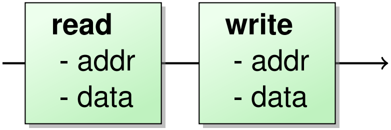
A high level of abstraction,
that appears early in the design-flow

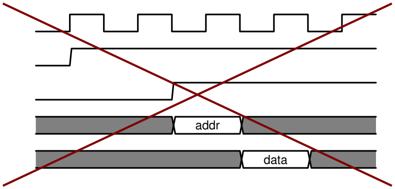
$\approx$ 1000 Faster than
low-level simulations (RTL)

In production in the industry

Content of a TLM Model

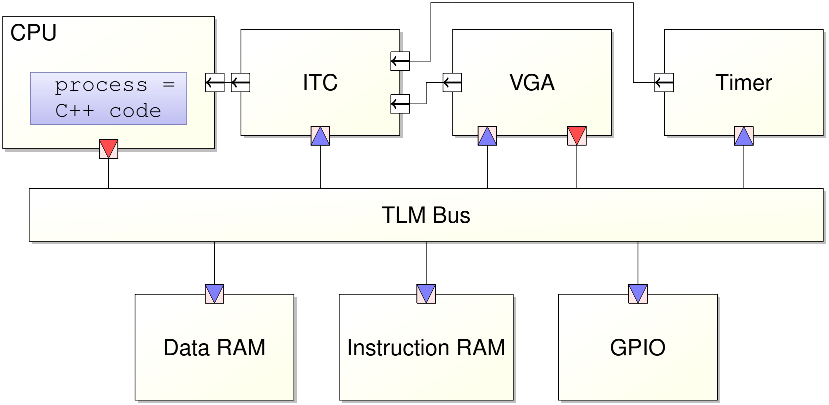
- Model what is needed for Software Execution:
  - Processors
  - Address-map
  - Concurrency
- ... and only that.
  - No micro-architecture
  - No bus protocol
  - No pipeline
  - No physical clock
  - . . .





Standard for TLM = SystemC (IEEE1666)

An Example TLM Model



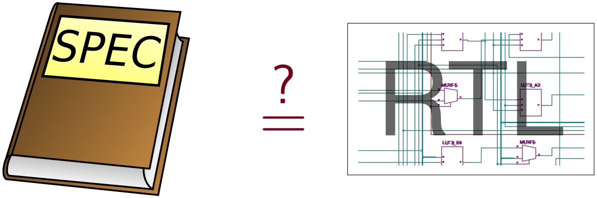
Uses of Functional Models

Reference for Hardware Validation

Virtual Prototype for Software Development

Uses of Functional Models

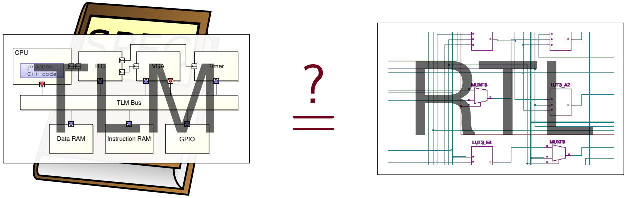
Reference for Hardware Validation



Virtual Prototype for Software Development

Uses of Functional Models

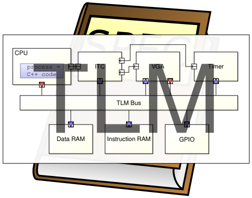
Reference for Hardware Validation



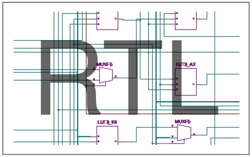
Virtual Prototype for Software Development

Uses of Functional Models

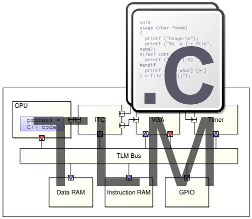
Reference for Hardware Validation



$\overset{?}{=}$



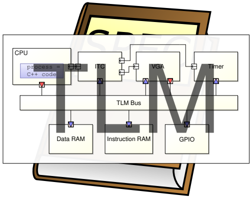
Virtual Prototype for Software Development



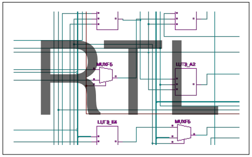
```
void
usage (char *name)
{
  printf ("Usage:\n");
  printf ("%s -a [-c file",
name);
#ifdef LOFI
  printf ("[illegible] [-G] ");
#endif
  printf ([illegible] what] [-r]
[-u file [illegible]]");
```

Uses of Functional Models

Reference for Hardware Validation

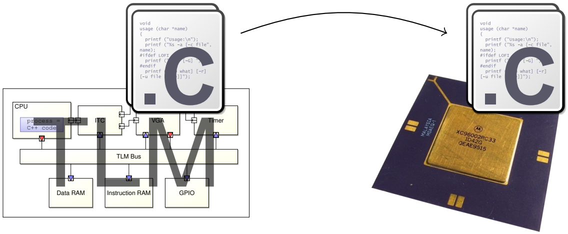


$\overset{?}{=}$



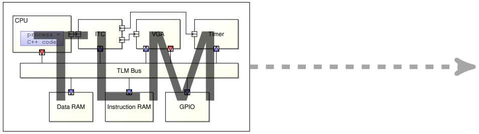
Unmodified Software

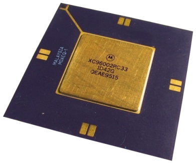
Virtual Prototype for Software Development



Non-Functional Models

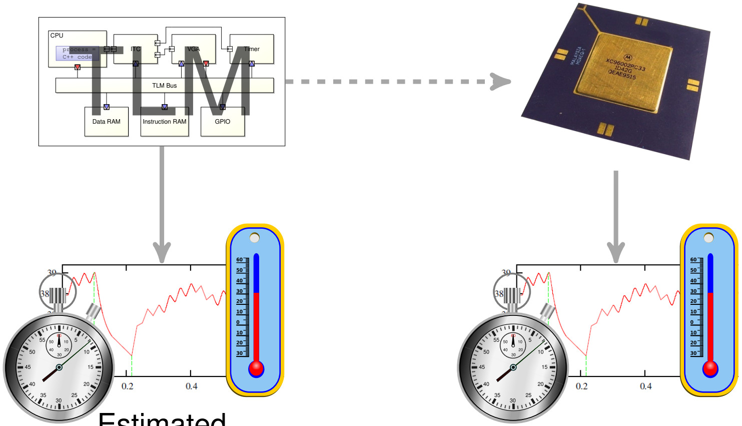
Timing, Power consumption, Temperature Estimation





Non-Functional Models

Timing, Power consumption, Temperature Estimation

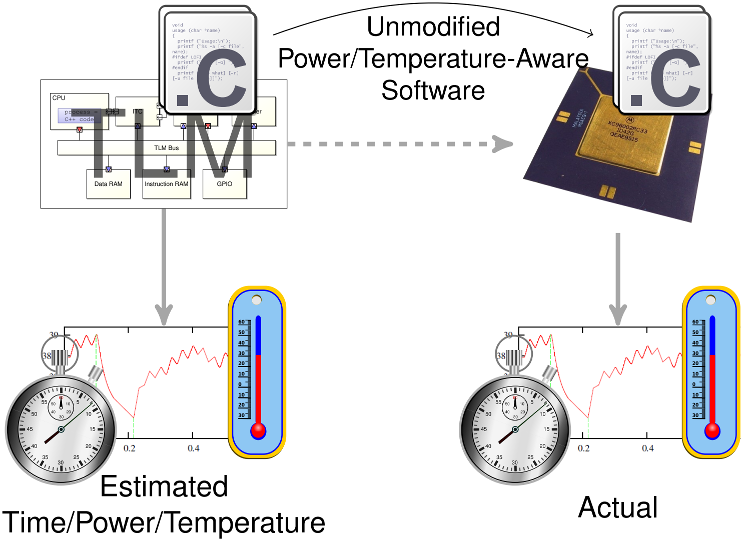


Estimated
Time/Power/Temperature

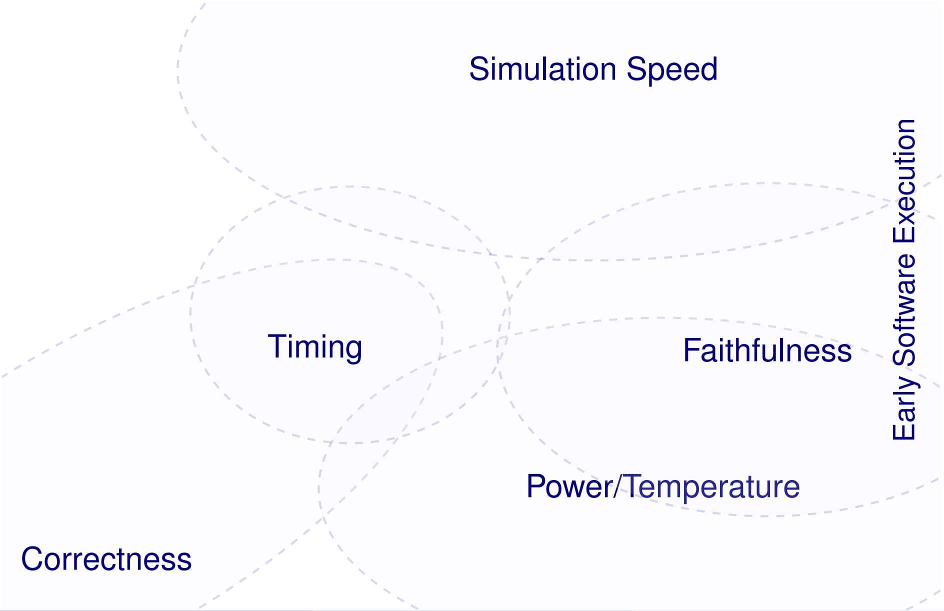
Actual

Non-Functional Models

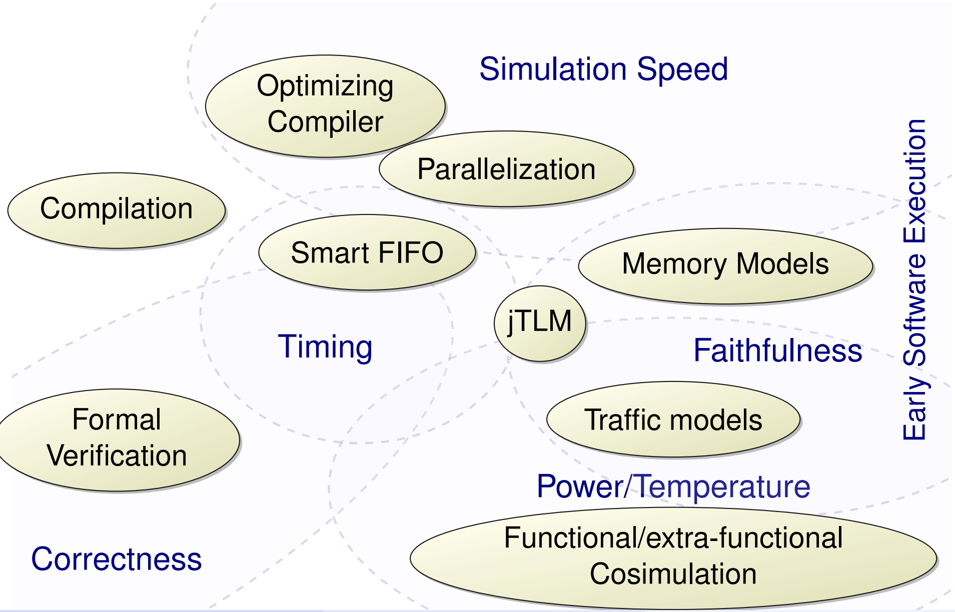
Timing, Power consumption, Temperature Estimation



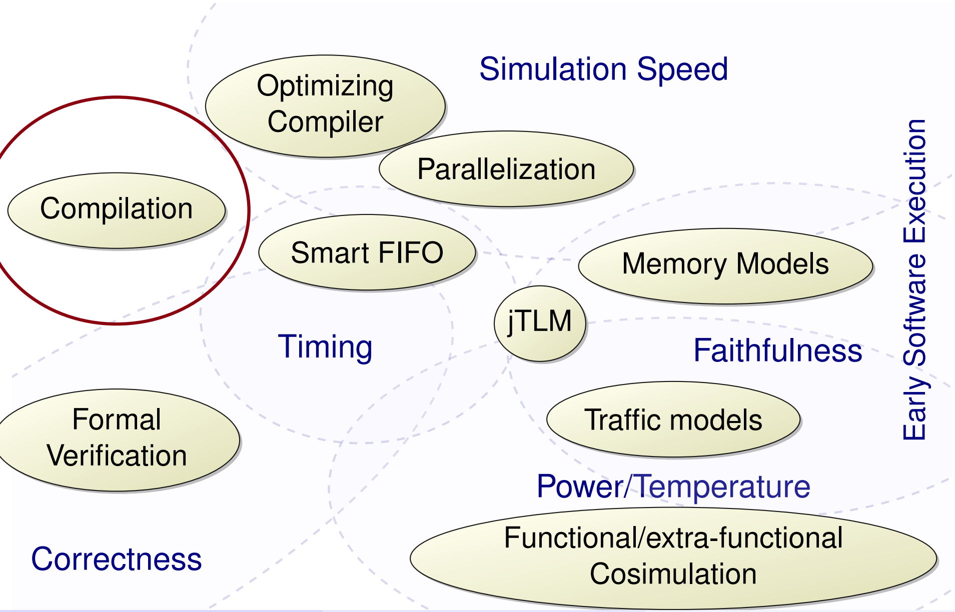
Contributions on TLM



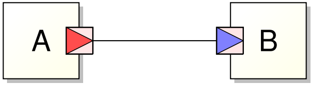
Contributions on TLM



Contributions on TLM



SystemC: Simple Example



```
int sc_main (int, char **) {
   /* Instanciate modules */
   A a("Alice");
   B b("Bob");

   /* Connect them together */
   a.socket.bind(b.socket);

   /* and start simulation */
   sc_start();
   return 0;
}
```

```
struct A : sc_module {
   /* Connection to outside */
   initiator_socket socket;

   /* Behavior */
   void thread() {
      do_stuff();
      write(socket, addr, data);
   }

   SC_CTOR(A) {
      SC_THREAD(thread);
   }
};
```

$\leadsto$ Compilable with any C++ compiler

Challenges and Solutions with SystemC Front-Ends

1. C++ is complex (e.g. `clang++` ≈ 400,000 LOC)

2. Architecture built at runtime, with C++ code

```
int sc_main (int, char **) {
   /* Instanciate modules */
   A a("Alice");
   B b("Bob");

   /* Connect them together */
   a.socket.bind(b.socket);

   /* and start simulation */
   sc_start();
   return 0;
}
```

```
struct A : sc_module {
   /* Connection to outside */
   initiator_socket socket;

   /* Behavior */
   void thread() {
      do_stuff();
      write(socket, addr, data);
   }

   SC_CTOR(A) {
      SC_THREAD(thread);
   }
};
```

Kevin Marquet

Guillaume Sergent

Challenges and Solutions with SystemC Front-Ends

1. C++ is complex (e.g. `clang++` ≈ 400,000 LOC)
   ⇝ Write a C++ front-end or reuse one (g++, clang++, EDG, . . .)
2. Architecture built at runtime, with C++ code
   ⇝ Analyze elaboration phase or execute it

```
int sc_main (int, char **) {
   /* Instanciate modules */
   A a("Alice");
   B b("Bob");

   /* Connect them together */
   a.socket.bind(b.socket);

   /* and start simulation */
   sc_start();
   return 0;
}
```

```
struct A : sc_module {
   /* Connection to outside */
   initiator_socket socket;

   /* Behavior */
   void thread() {
      do_stuff();
      write(socket, addr, data);
   }

   SC_CTOR(A) {
      SC_THREAD(thread);
   }
};
```

Kevin Marquet

Guillaume Sergent

Challenges and Solutions with SystemC Front-Ends

1. C++ is complex (e.g. `clang++` ≈ 400,000 LOC)
   $\leadsto$ Write a C++ front-end or reuse one (g++, clang++, EDG, ...)
2. Architecture built at runtime, with C++ code
   $\leadsto$ Analyze elaboration phase or execute it

```
int sc_main (int, char **) {
    /* Instan  iate modules */
    A a("Ali  ");
    B b("Bob

              ther */
              ;

              n */

    return 0;
}
```

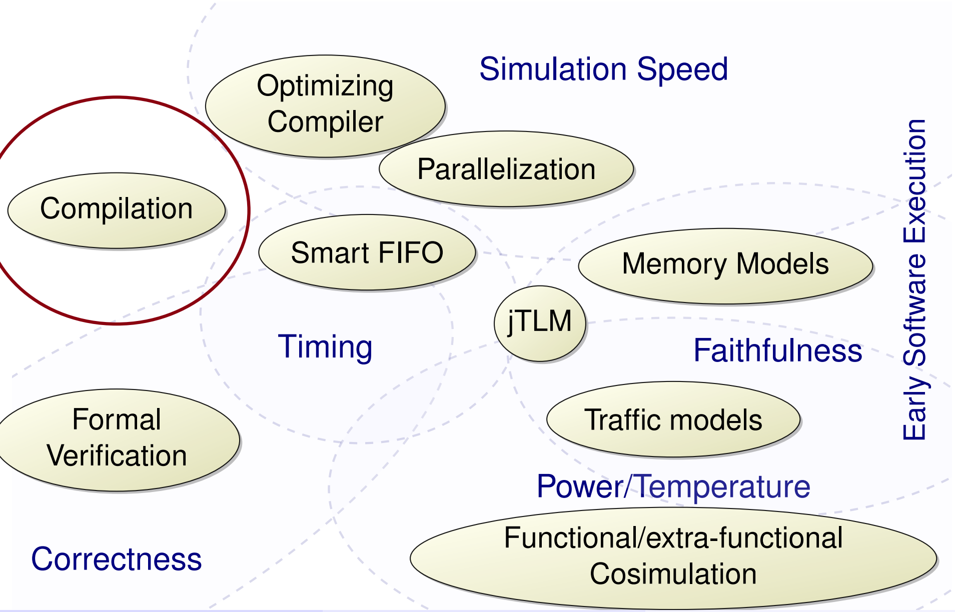
Static Approaches

```
struct A : s  module {
    /* Connect     to outside */
    initiato

    /* Be
    void
        do_st
        write(socke         data);
    }

    SC_CTOR(A) {
        SC_THREAD(thread);
    }
};
```

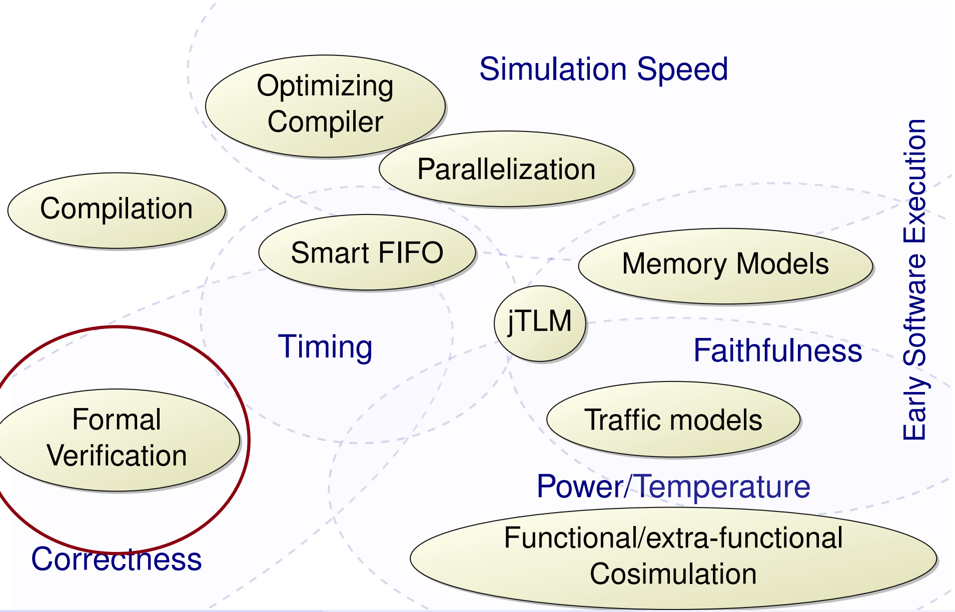
Dynamic Approaches

Kevin Marquet Guillaume Sergent

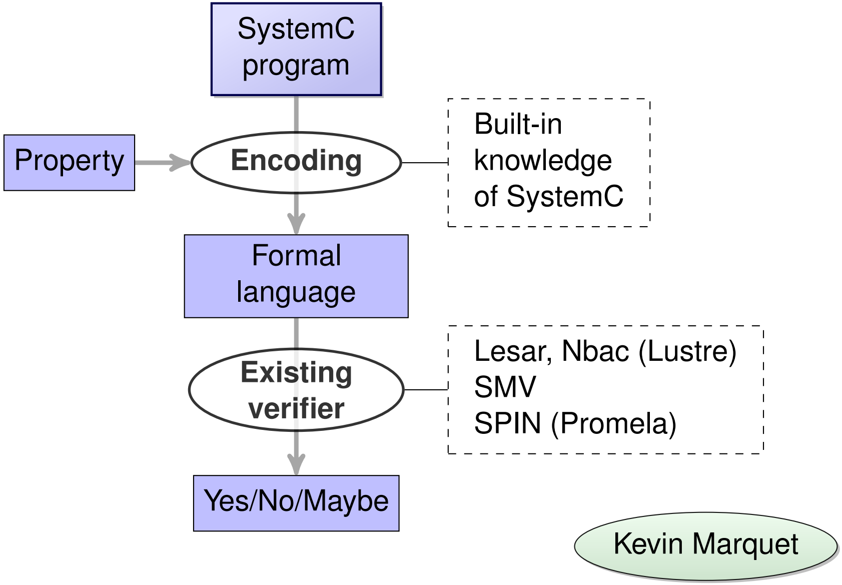
Contributions on TLM



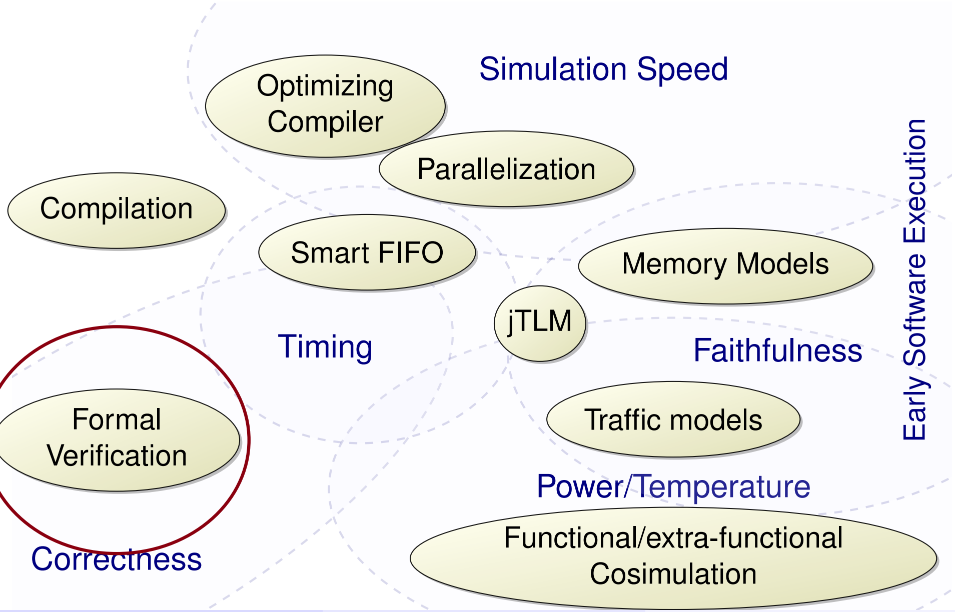
Contributions on TLM



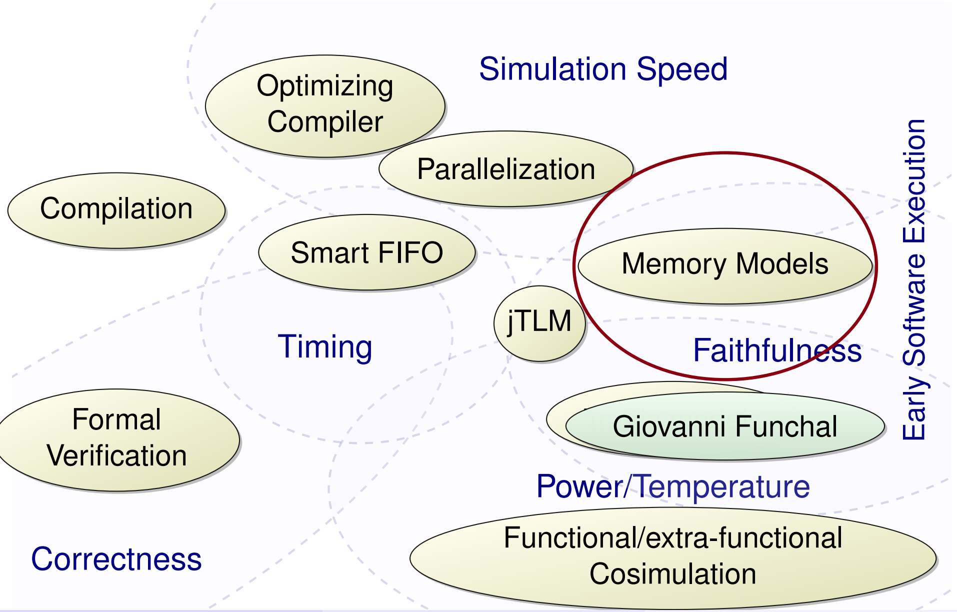
Formal Verification: Encoding Approaches



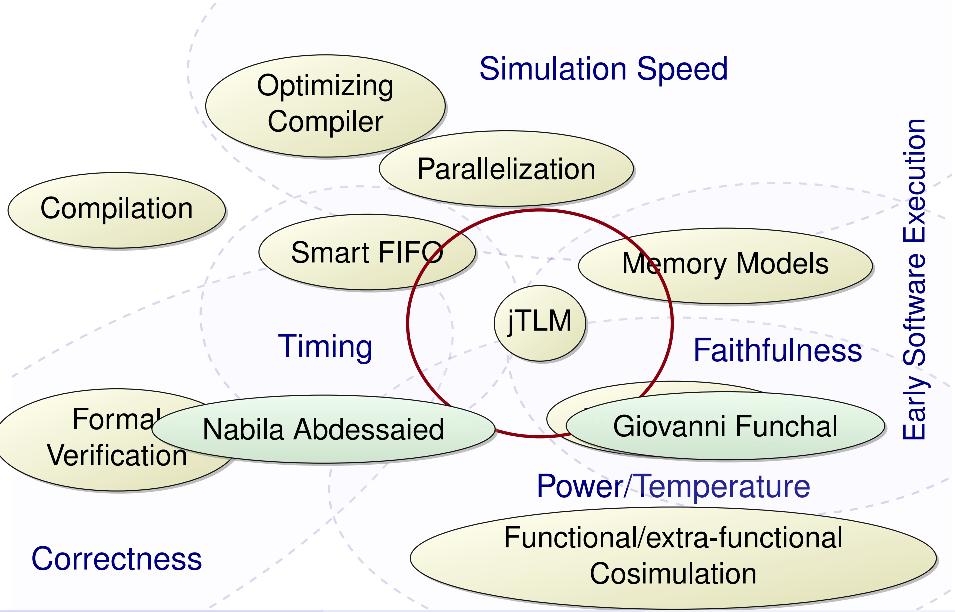
Contributions on TLM



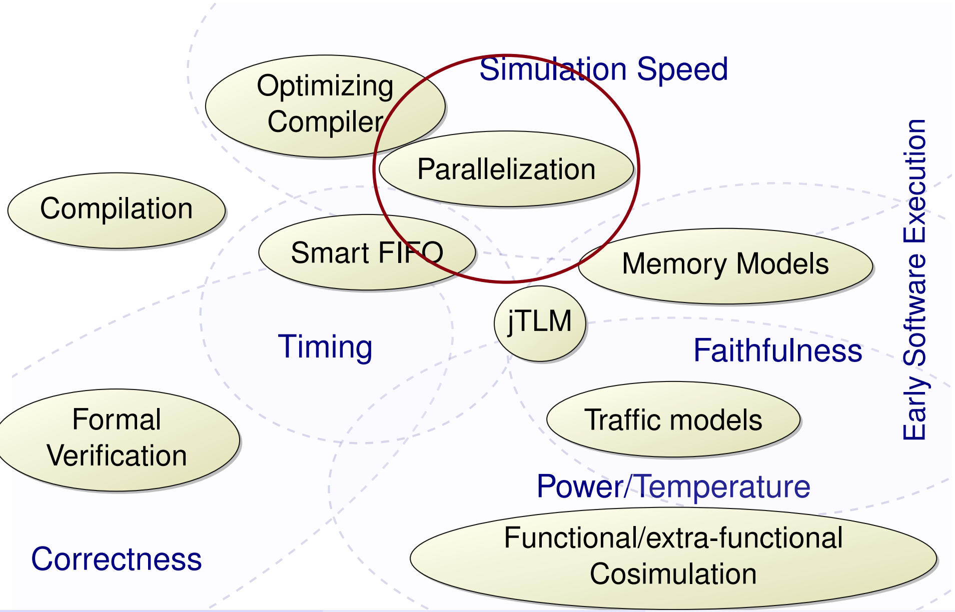
Contributions on TLM



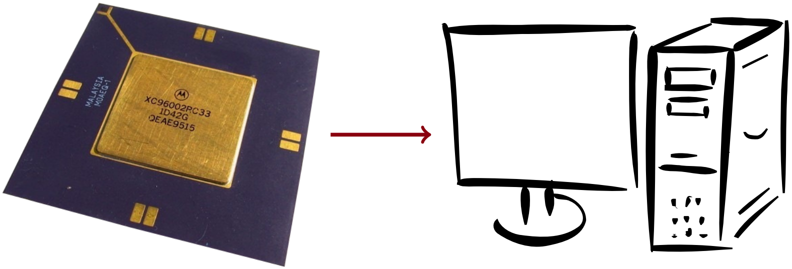
Contributions on TLM



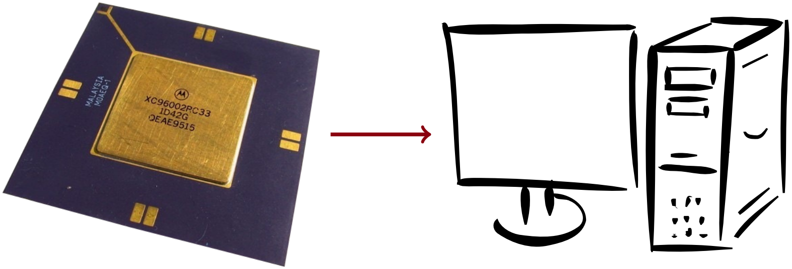
Contributions on TLM



Simulation Parallelization



Simulation Parallelization



SystemC uses co-routine semantics
(Sequential)

Problems and Solutions for Parallel Execution of SystemC/TLM

(1) Execution order imposed by SystemC

(2) Race conditions (e.g. `x++` on global variable)

Problems and Solutions for Parallel Execution of SystemC/TLM

(1) Execution order imposed by SystemC

(2) Race conditions (e.g. `x++` on global variable)

$\leadsto$ No efficient and automatic solution for SystemC/TLM

Problems and Solutions for Parallel Execution of SystemC/TLM

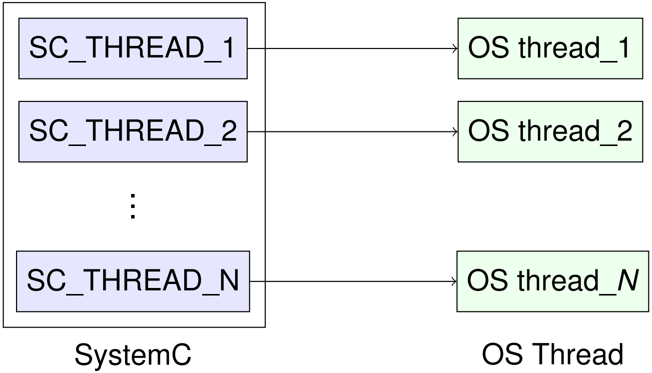
(1) Execution order imposed by SystemC

(2) Race conditions (e.g. `x++` on global variable)

$\leadsto$ No efficient and automatic solution for SystemC/TLM

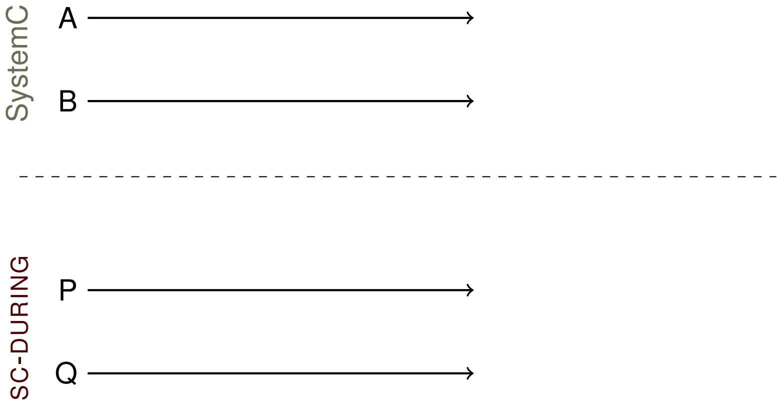
Our proposal = new constructs:
Desynchronisation (1) / Synchronisation (2)

SC-DURING: The Idea

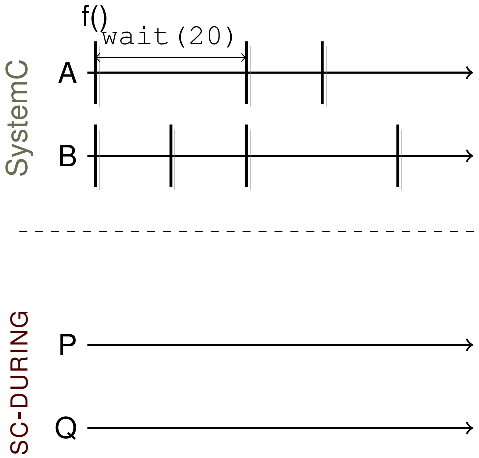


- Unmodified SystemC
- Computations delegated to external threads
- Weak synchronization between SystemC and tasks with duration

Simulated Time in SystemC and SC-DURING



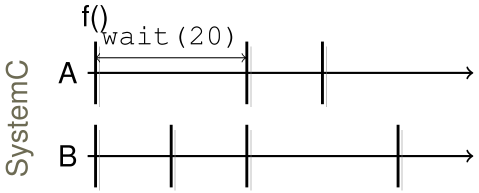
Simulated Time in SystemC and SC-DURING



Process A:

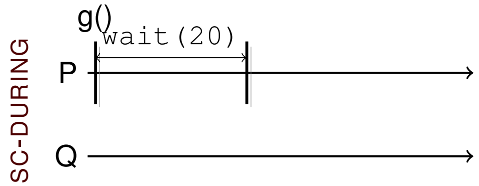
```
//computation
f();
//time taken by f
wait(20, SC_NS);
```

Simulated Time in SystemC and SC-DURING



Process A:

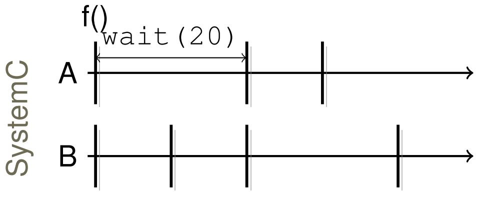
```
//computation
f();
//time taken by f
wait(20, SC_NS);
```



Process P:

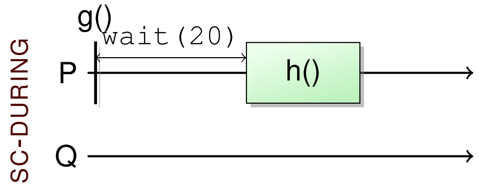
```
g();
wait(20, SC_NS);
```

Simulated Time in SystemC and SC-DURING



Process A:

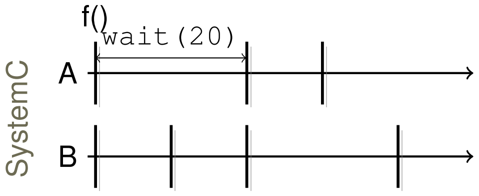
```
//computation
f();
//time taken by f
wait(20, SC_NS);
```



Process P:

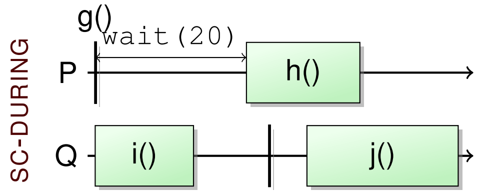
```
g();
wait(20, SC_NS);
during(15, SC_NS, h);
```

Simulated Time in SystemC and SC-DURING



Process A:

```
//computation
f();
//time taken by f
wait(20, SC_NS);
```

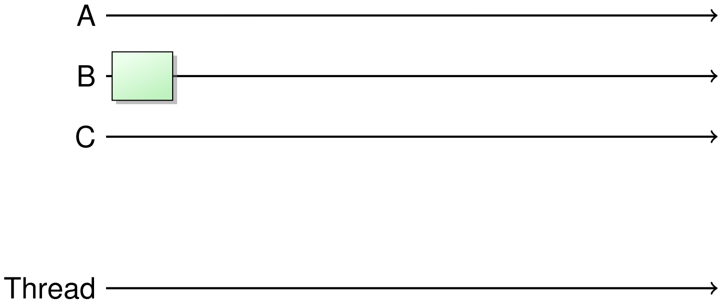


Process P:

```
g();
wait(20, SC_NS);
during(15, SC_NS, h);
```

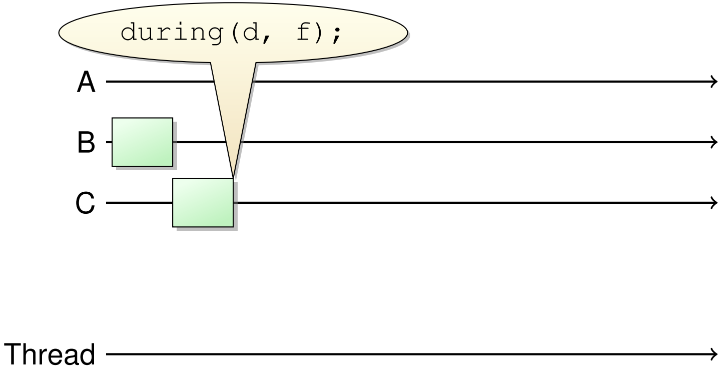
SC-DURING: Sketch of Implementation

```
void during(sc_core::sc_time d,
            std::function<void()> f) {
1   std::thread t(f); // Create thread
2   sc_core::wait(d); // SystemC executes
3   t.join(); // Wait for completion
}
```



SC-DURING: Sketch of Implementation

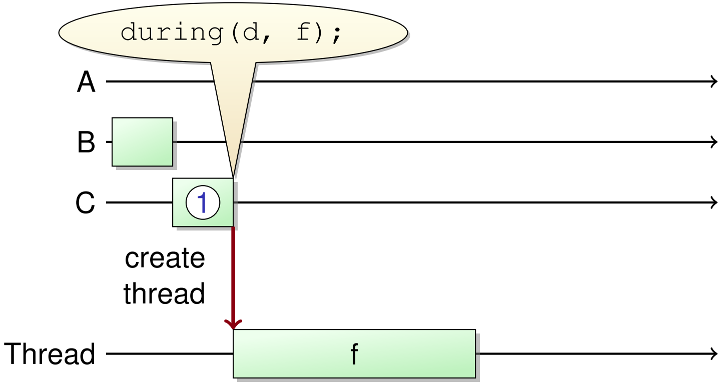
```
void during(sc_core::sc_time d,
            std::function<void()> f) {
①    std::thread t(f); // Create thread
②    sc_core::wait(d); // SystemC executes
③    t.join(); // Wait for completion
}
```



SC-DURING: Sketch of Implementation

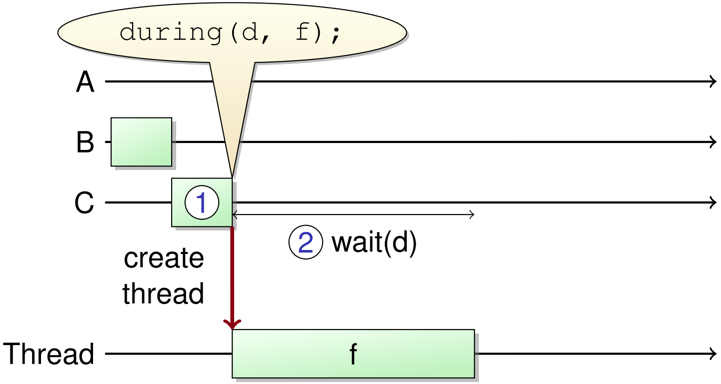
```
void during(sc_core::sc_time d,
            std::function<void()> f) {
    std::thread t(f); // Create thread
    sc_core::wait(d); // SystemC executes
    t.join(); // Wait for completion
}
```

Markers 1, 2, 3 stand beside the three lines of the function body.



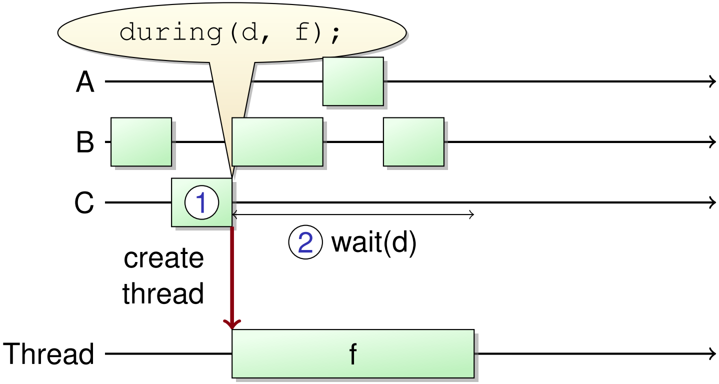
SC-DURING: Sketch of Implementation

```
    void during(sc_core::sc_time d,
                std::function<void()> f) {
(1)     std::thread t(f); // Create thread
(2)     sc_core::wait(d); // SystemC executes
(3)     t.join(); // Wait for completion
    }
```



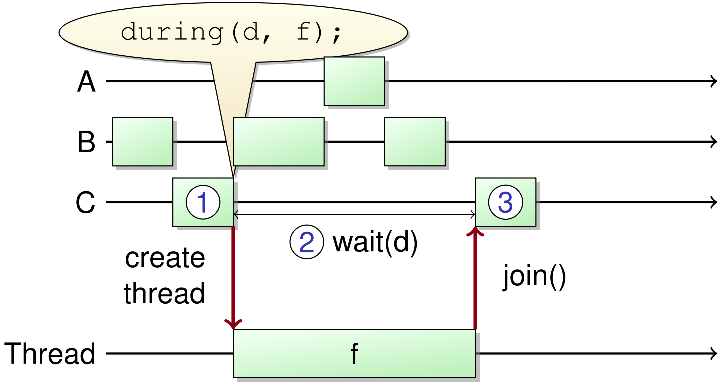
SC-DURING: Sketch of Implementation

```
void during(sc_core::sc_time d,
            std::function<void()> f) {
①   std::thread t(f); // Create thread
②   sc_core::wait(d); // SystemC executes
③   t.join(); // Wait for completion
}
```



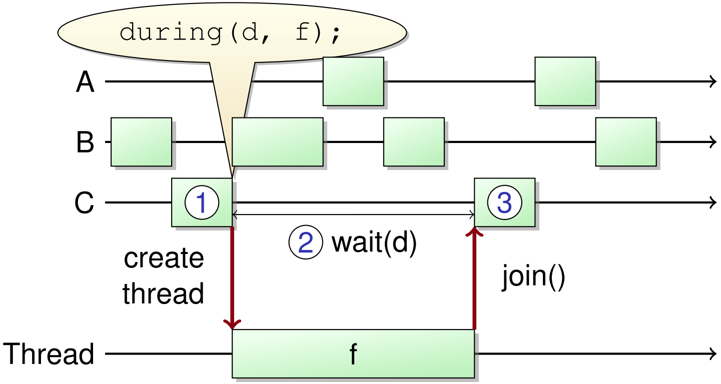
SC-DURING: Sketch of Implementation

```
  void during(sc_core::sc_time d,
              std::function<void()> f) {
1     std::thread t(f); // Create thread
2     sc_core::wait(d); // SystemC executes
3     t.join(); // Wait for completion
  }
```



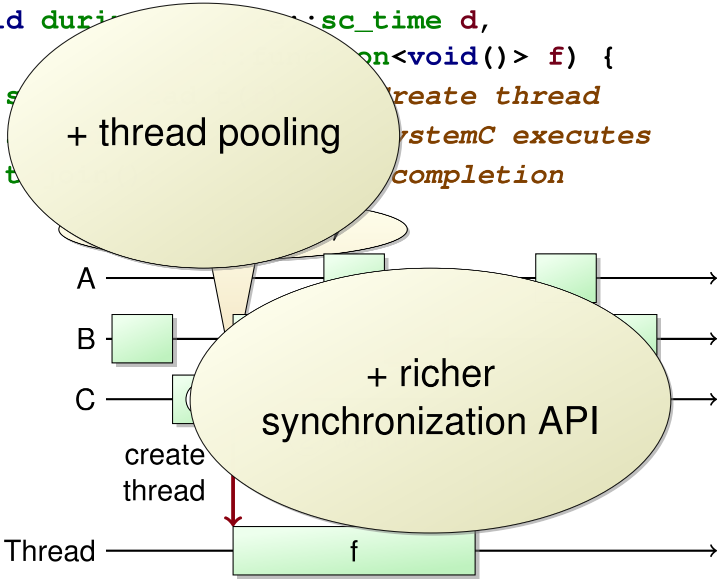
SC-DURING: Sketch of Implementation

```
void during(sc_core::sc_time d,
            std::function<void()> f) {
①   std::thread t(f); // Create thread
②   sc_core::wait(d); // SystemC executes
③   t.join(); // Wait for completion
}
```

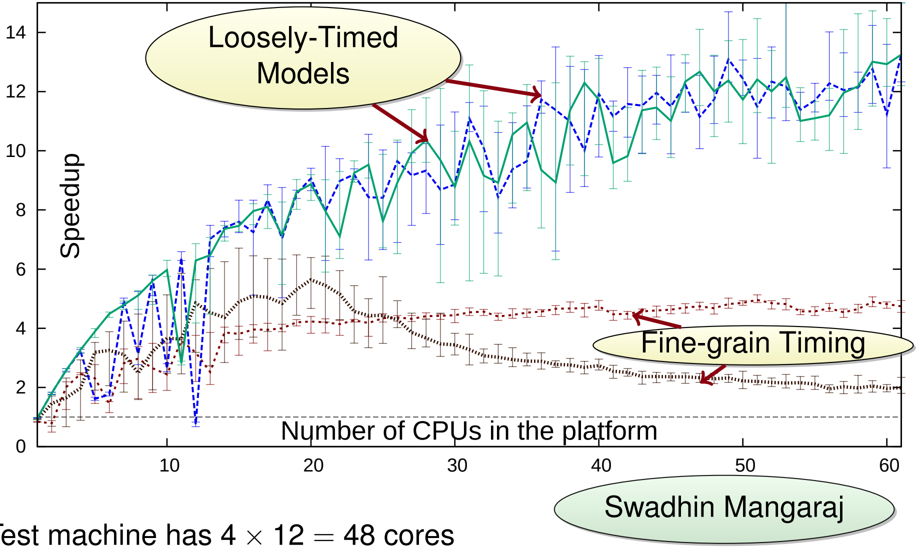


SC-DURING: Sketch of Implementation

```
    void duri          ::sc_time d,
                         on<void()> f) {
1       s                 reate thread
2                         ystemC executes
3       t                 completion
    }
```

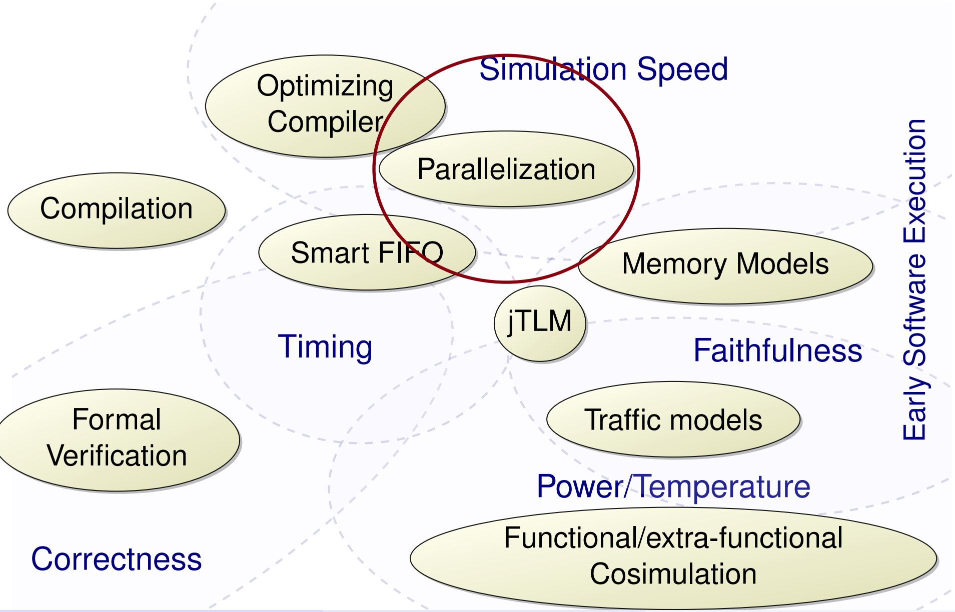


SC-DURING: Results

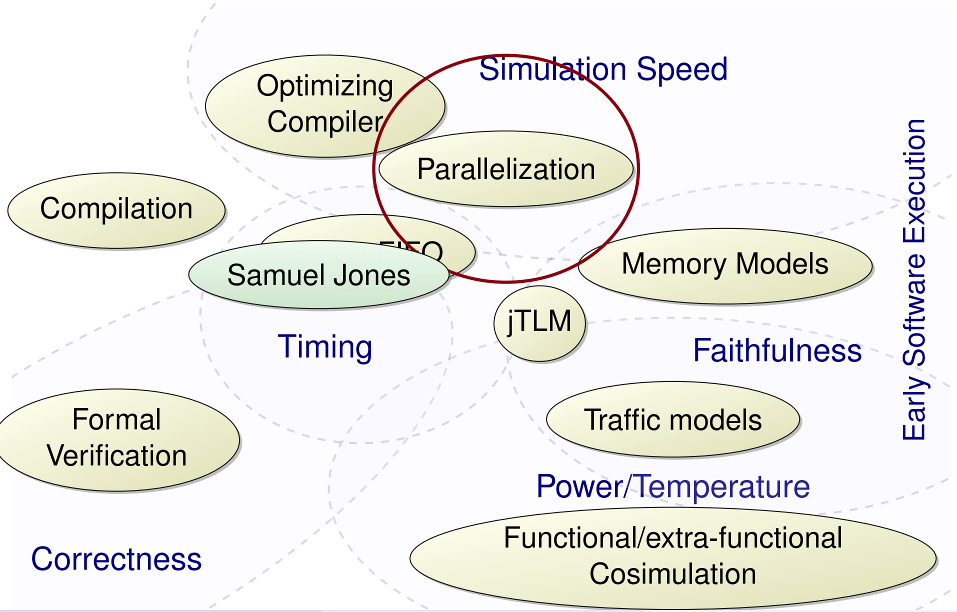


Test machine has $4 \times 12 = 48$ cores

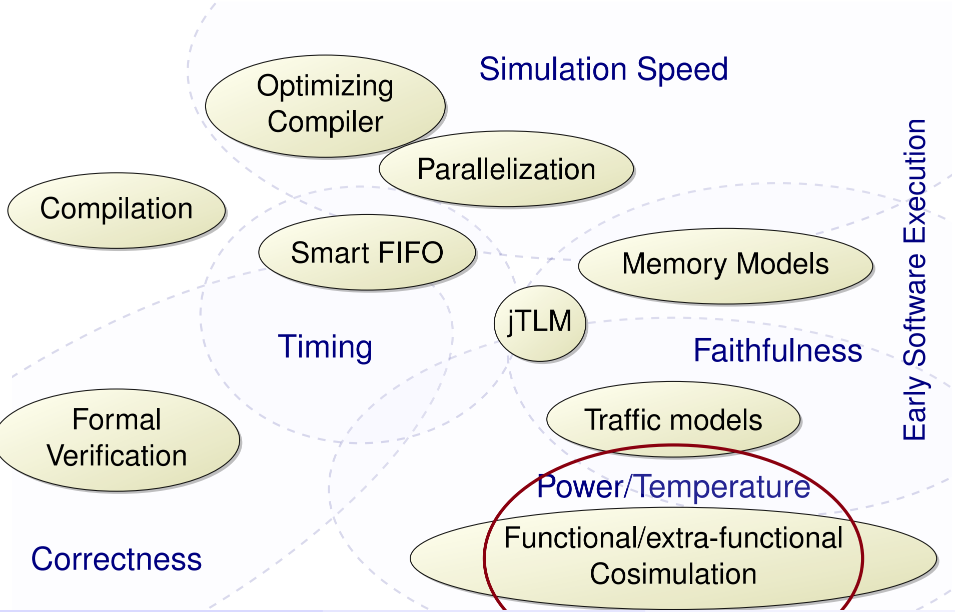
Contributions on TLM



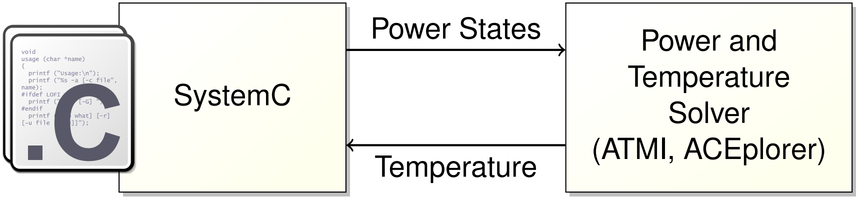
Contributions on TLM



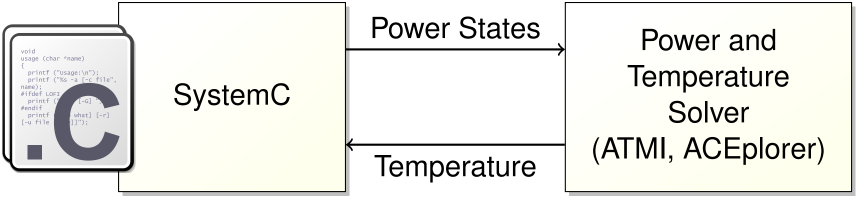
Contributions on TLM

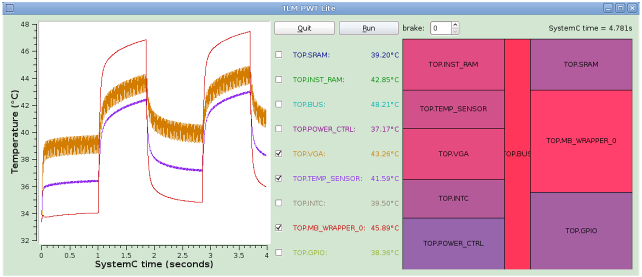


SystemC and Extra-Functional Solver Cosimulation



SystemC and Extra-Functional Solver Cosimulation



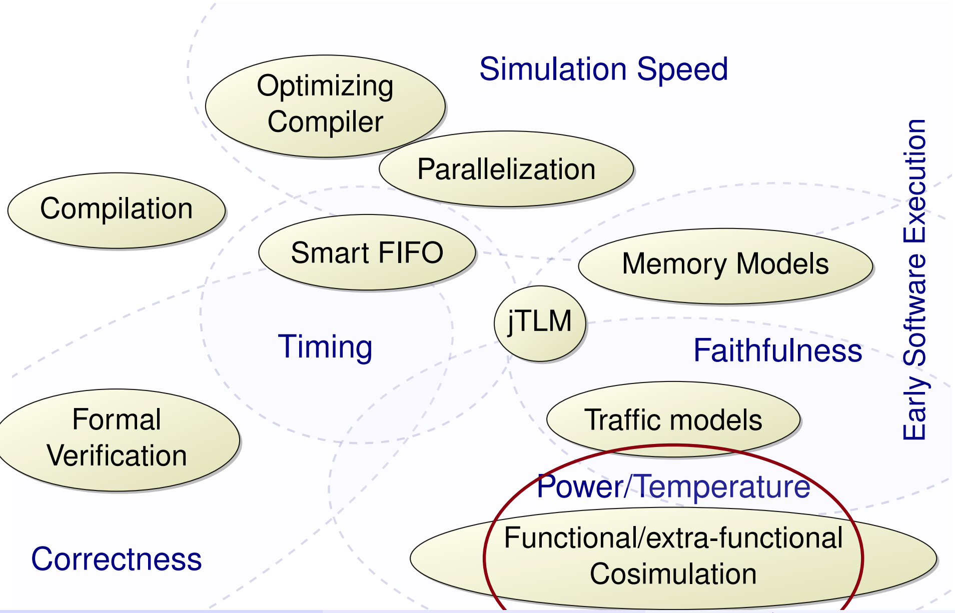

TLM PWT Lite

Quit   Run   brake: 0   SystemC time = 4.781s

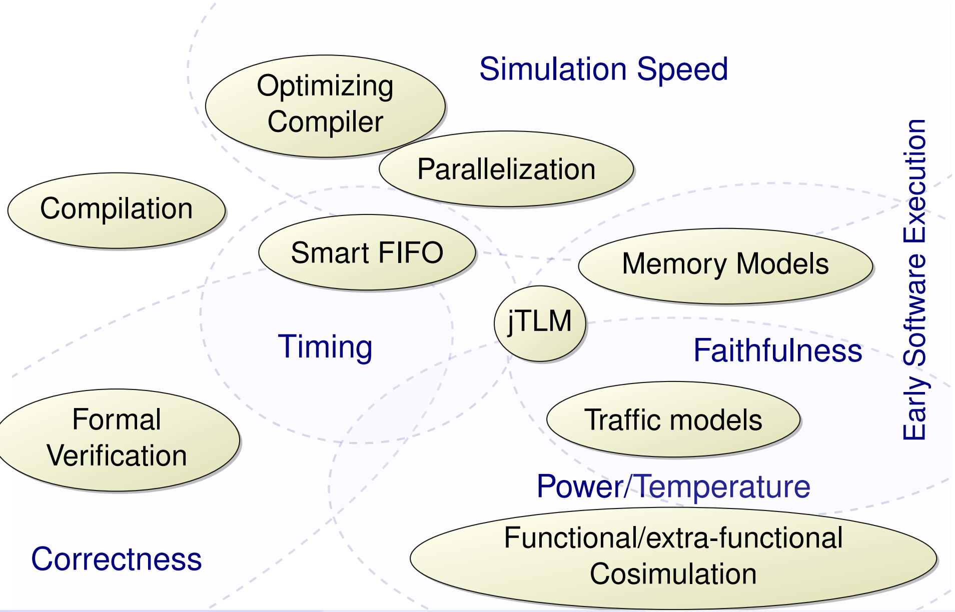
- [ ] TOP.SRAM: 39.20°C
- [ ] TOP.INST_RAM: 42.85°C
- [ ] TOP.BUS: 48.21°C
- [ ] TOP.POWER_CTRL: 37.17°C
- [x] TOP.VGA: 43.26°C
- [x] TOP.TEMP_SENSOR: 41.59°C
- [ ] TOP.INTC: 39.50°C
- [x] TOP.MB_WRAPPER_0: 45.89°C
- [ ] TOP.GPIO: 38.36°C

Claude Helmstetter   Tayeb Bouhadiba

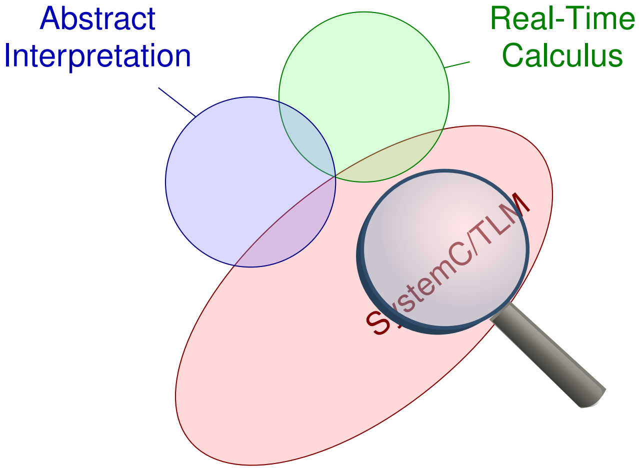
Contributions on TLM



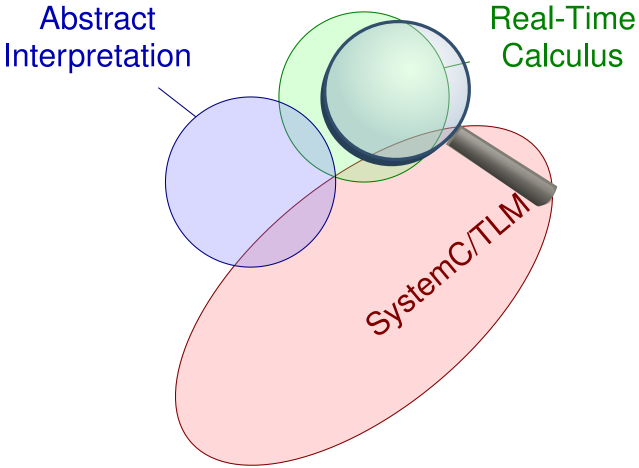
Contributions on TLM



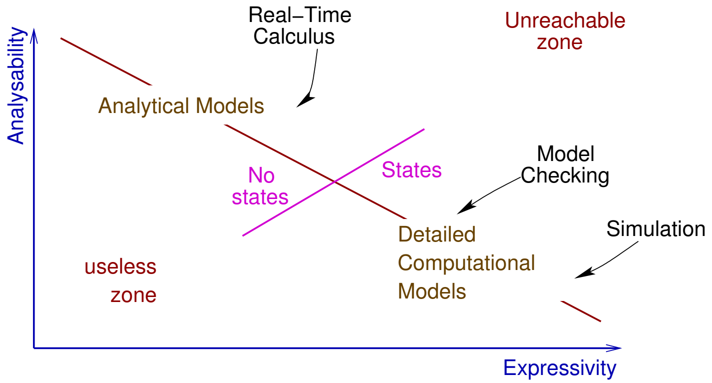
Outline



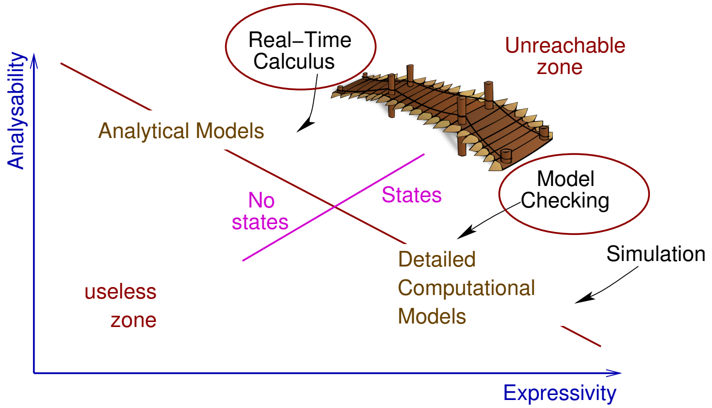
Outline



Precision Vs Algorithmic Complexity

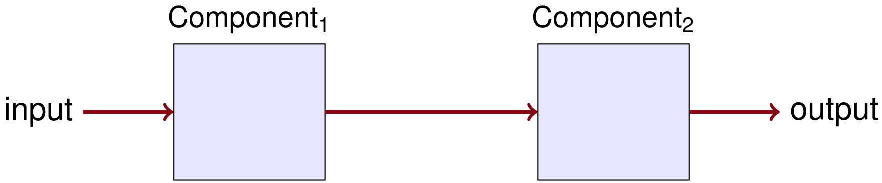


Precision Vs Algorithmic Complexity



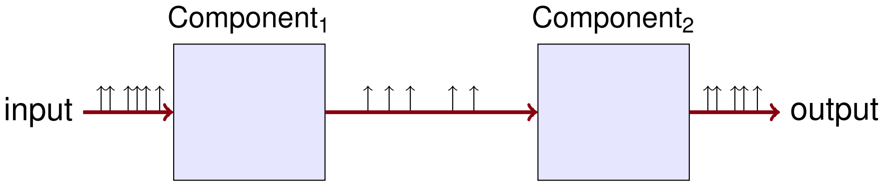
Modular Performance Analysis (MPA): The Big Picture

[Thiele et al.]



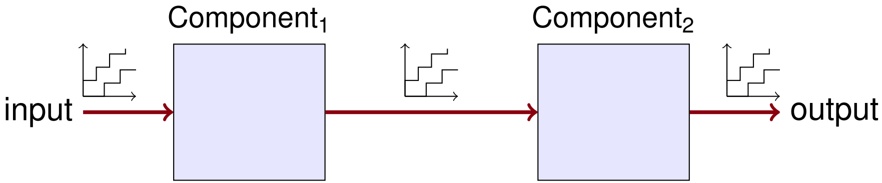
Modular Performance Analysis (MPA): The Big Picture

[Thiele et al.]



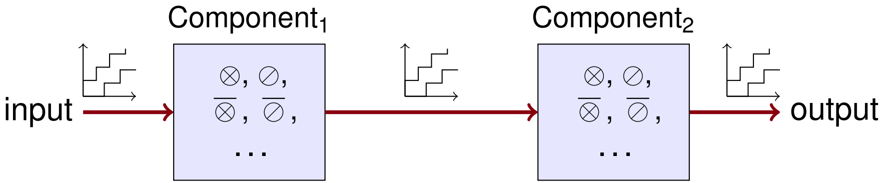
Modular Performance Analysis (MPA): The Big Picture

[Thiele et al.]

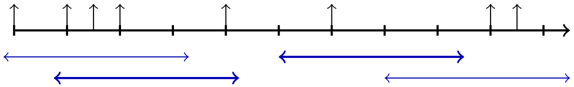


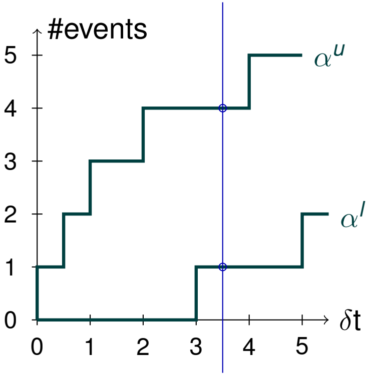
Modular Performance Analysis (MPA): The Big Picture

[Thiele et al.]



Real-Time Calculus and Arrival Curves

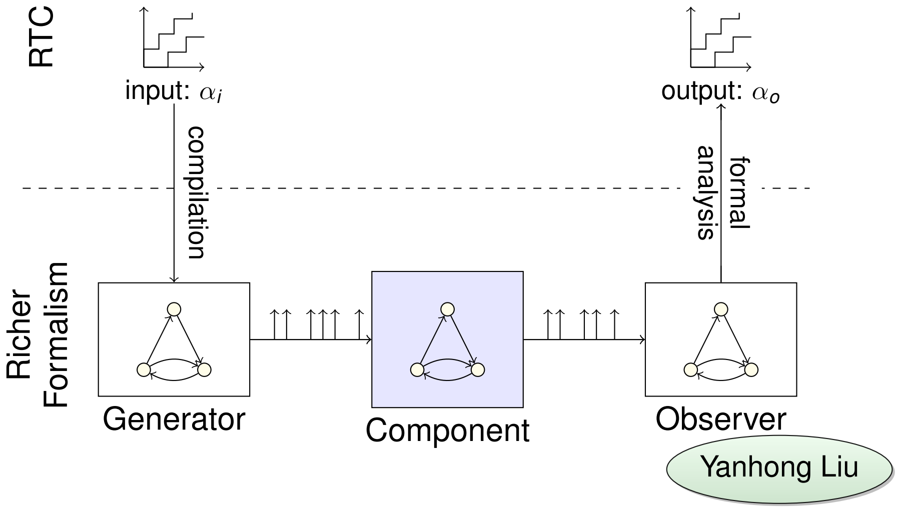




$\alpha^u(t)$ / $\alpha^l(t)$: min/max number of events in any window of duration t.

Interfacing with Timed Automata

[Chakraborty et al.]

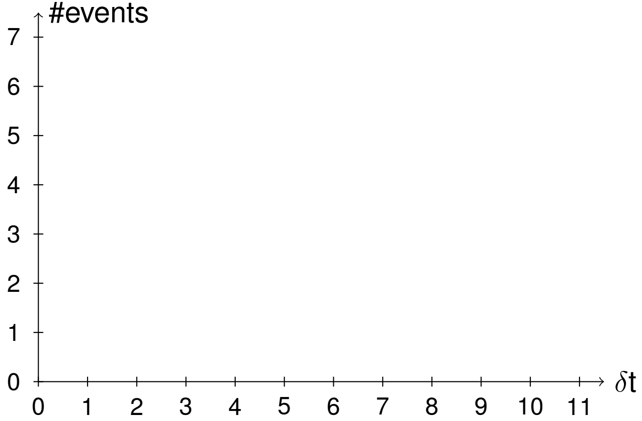


Timed Automata Vs Abstract Interpretation and SMT-Solving

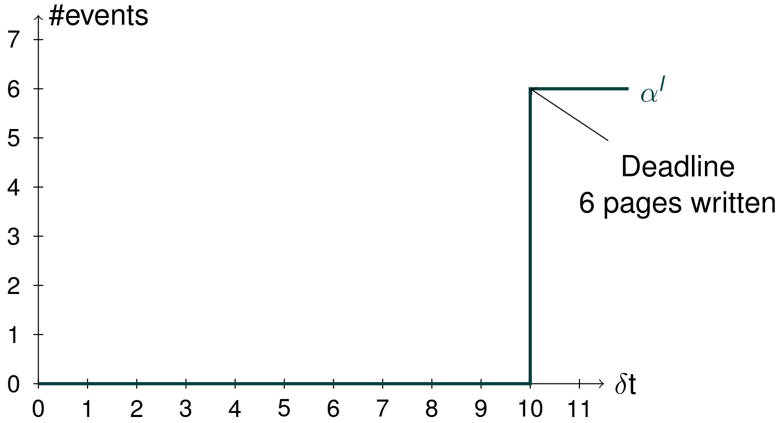
	Large event counters	Large timing constants
Timed automata (Uppaal)	💣	🍰
SMT solving	🍰	💣
Abstract Interpretation	🍰	💣

ac2lus: use Lustre tools to analyze MPA components
(Nbac = abstract interpretation, Kind = SMT solving)

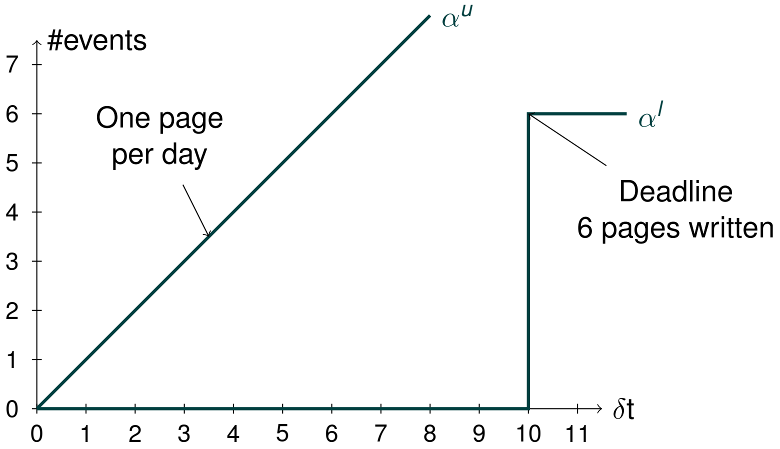
The Causality Problem



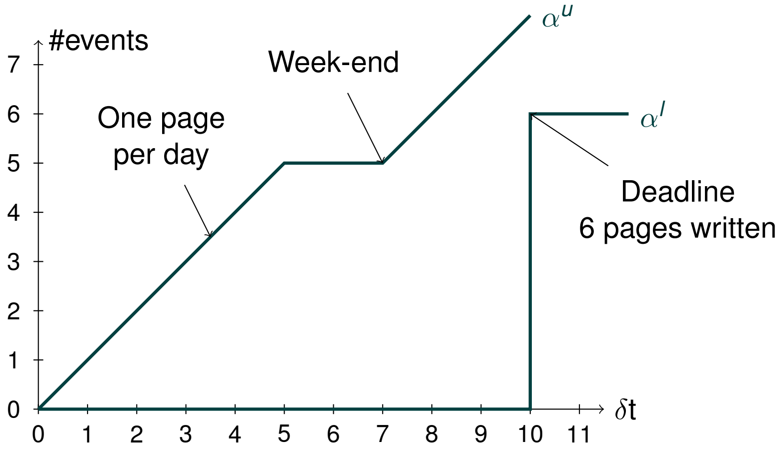
The Causality Problem



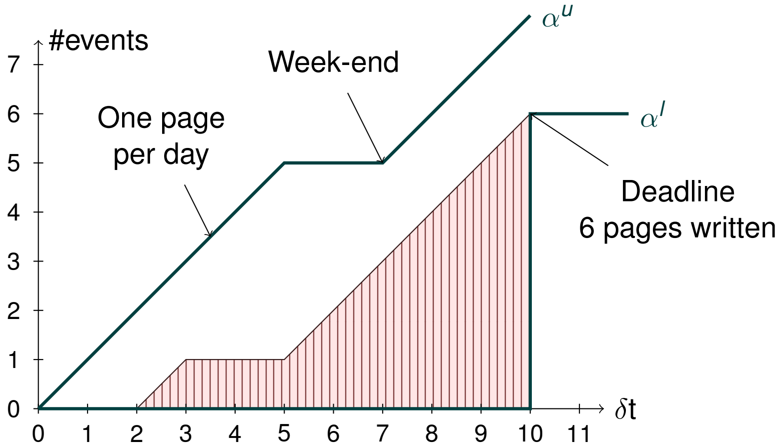
The Causality Problem



The Causality Problem

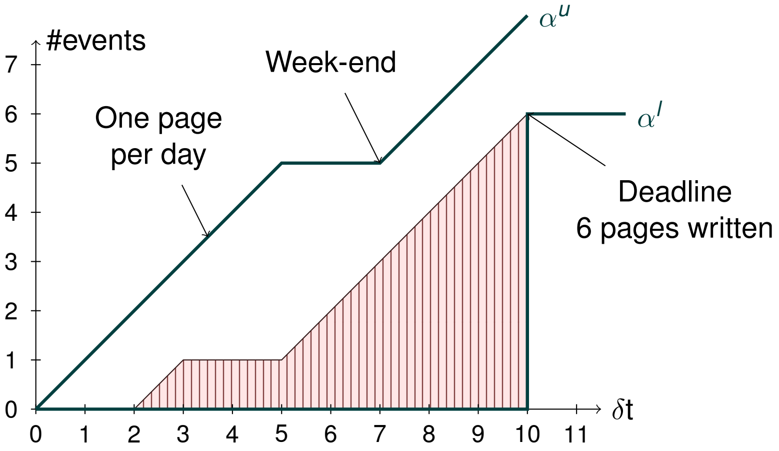


The Causality Problem



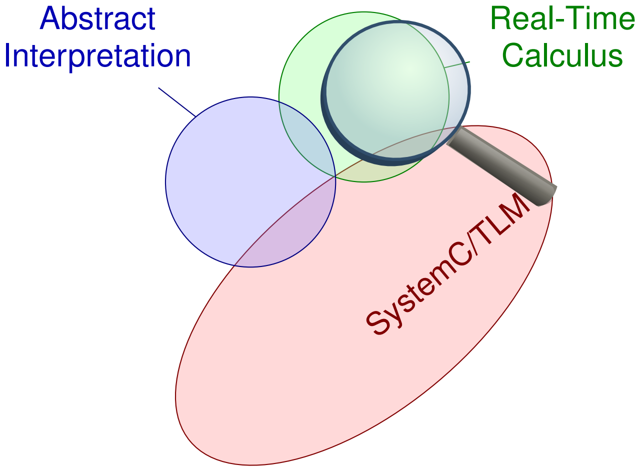
Implicit constraint: maximal procrastination = 2 days

The Causality Problem

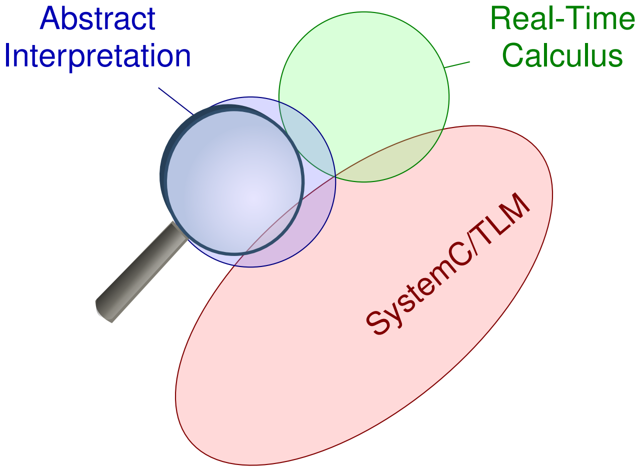


Causality closure = compute the implicit constraint automatically

Outline



Outline



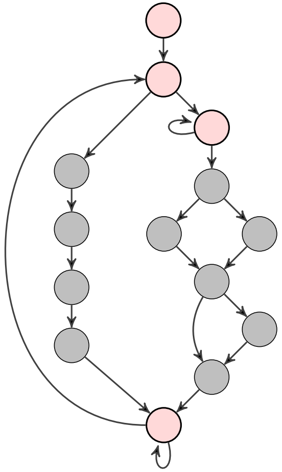
Abstract Interpretation and PAGAI

```
$ pagai -i test.c
void f() {
        int x = 0, y = 1;
        /* invariant:
           y = x + 1
           y <= 102
           y >= 1
        */
        while (x <= 100) {
                x++;
                y++;
        }
}
```

Julien Henry

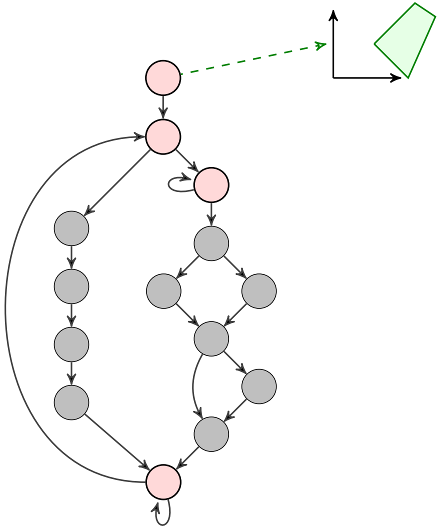
Abstract Interpretation with PAGAI

[Cousot & Cousot]



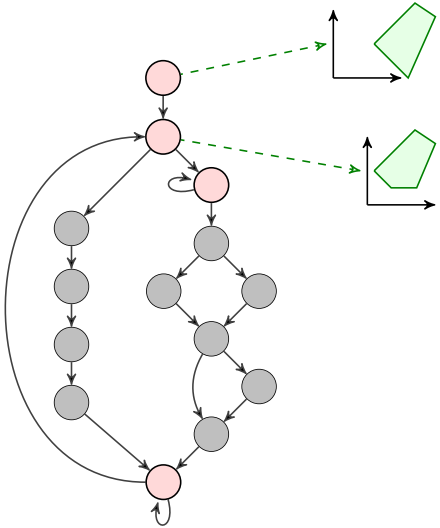
Abstract Interpretation with PAGAI

[Cousot & Cousot]



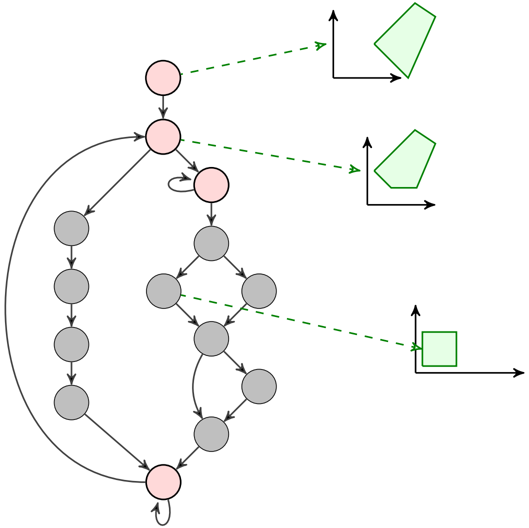
Abstract Interpretation with PAGAI

[Cousot & Cousot]



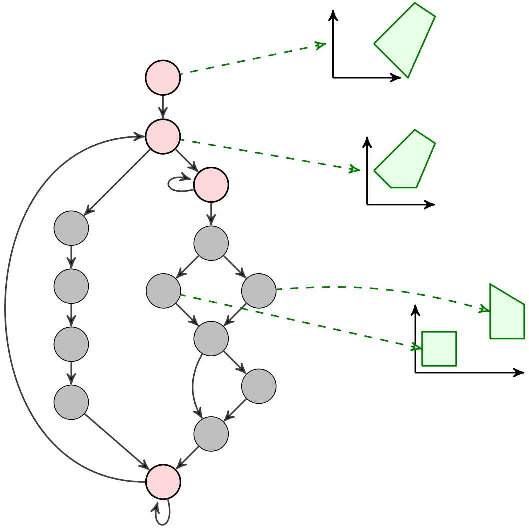
Abstract Interpretation with PAGAI

[Cousot & Cousot]



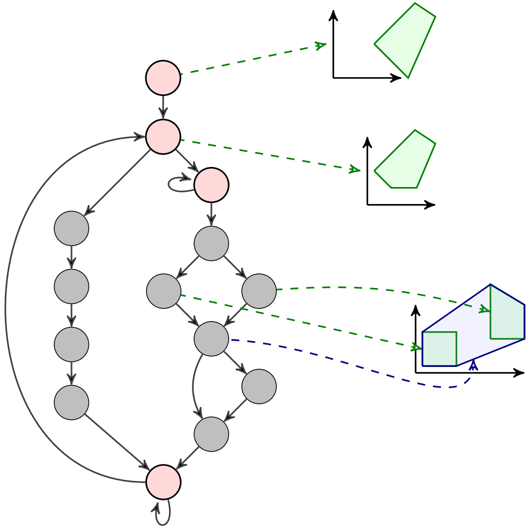
Abstract Interpretation with PAGAI

[Cousot & Cousot]



Abstract Interpretation with PAGAI

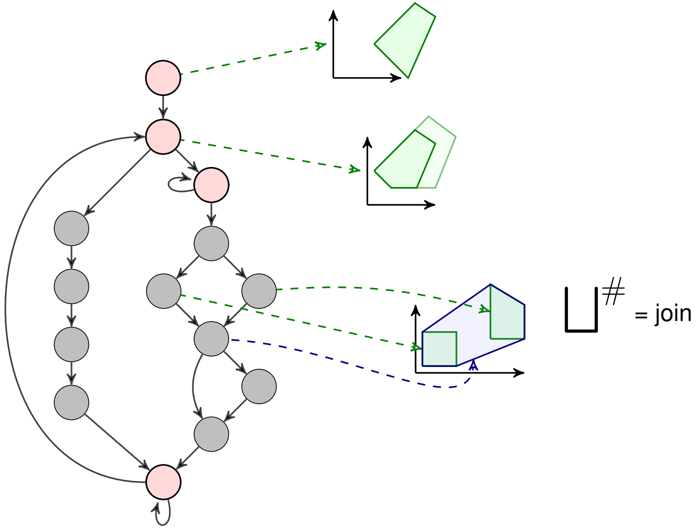
[Cousot & Cousot]



$\bigsqcup^{\#}$ = join

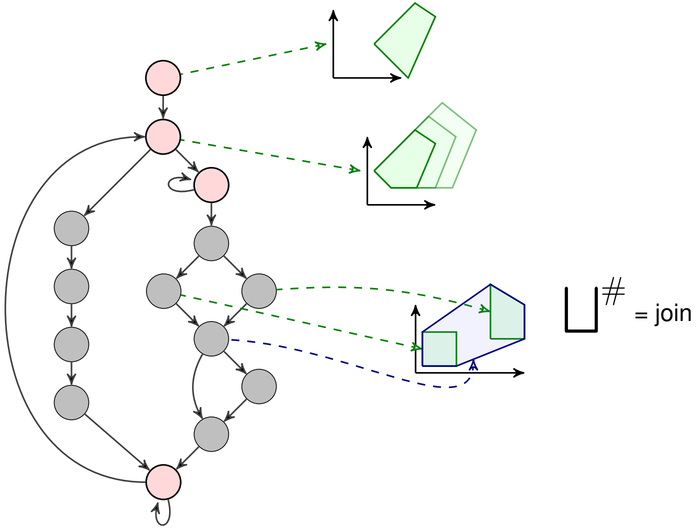
Abstract Interpretation with PAGAI

[Cousot & Cousot]



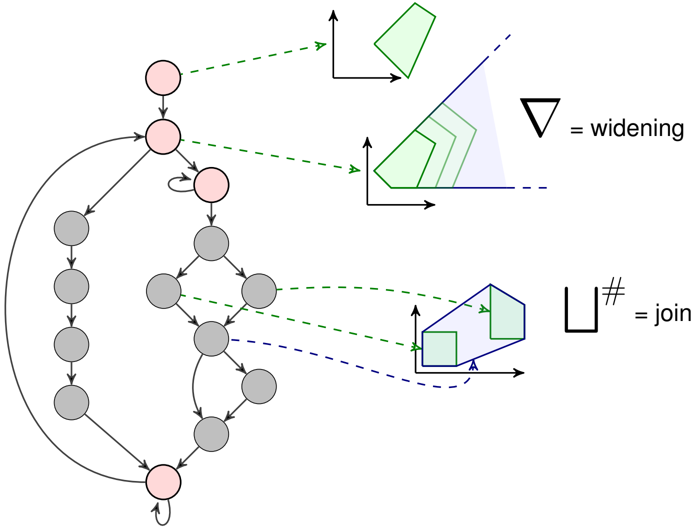
Abstract Interpretation with PAGAI

[Cousot & Cousot]



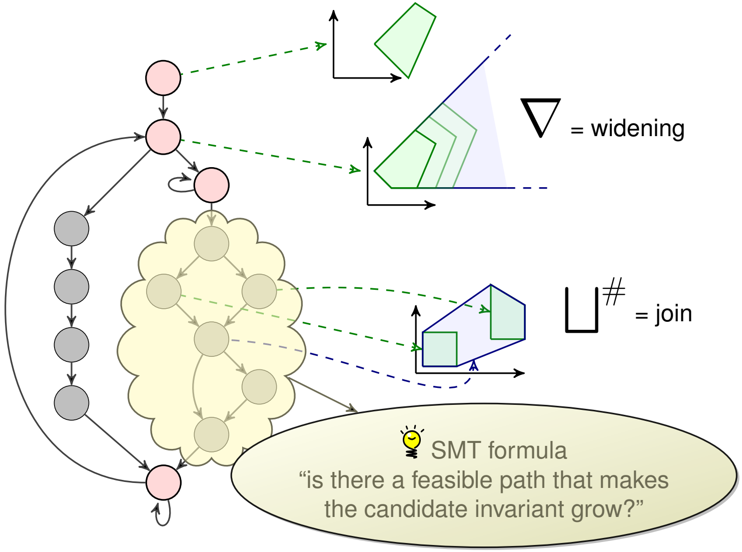
Abstract Interpretation with PAGAI

[Cousot & Cousot]



Abstract Interpretation with PAGAI

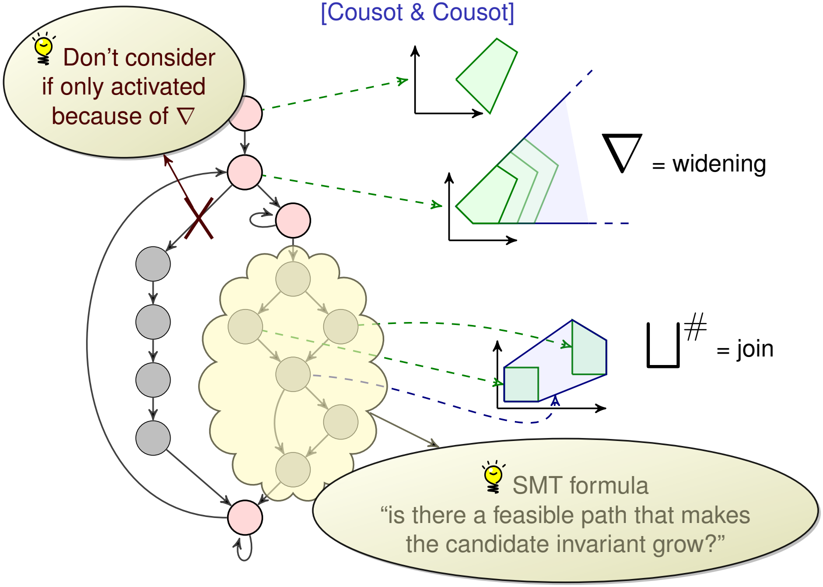
[Cousot & Cousot]



SMT formula
"is there a feasible path that makes the candidate invariant grow?"

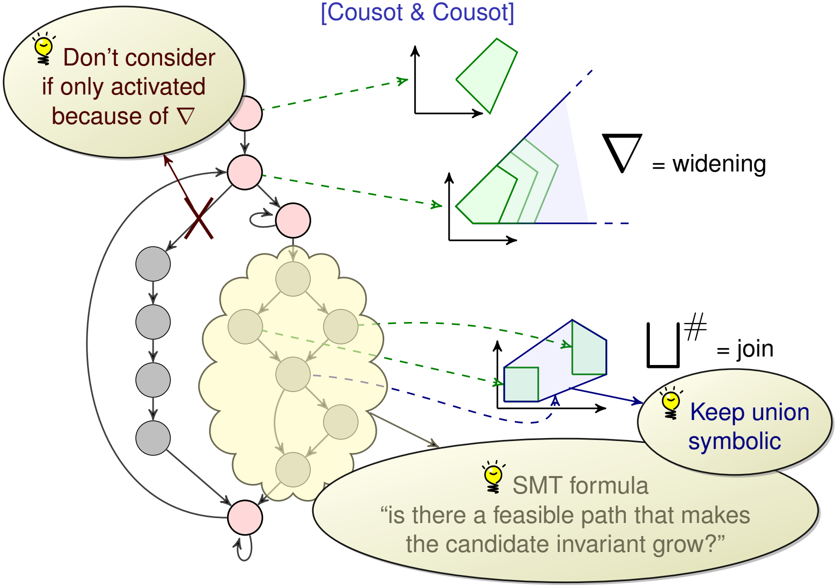
Abstract Interpretation with PAGAI

[Cousot & Cousot]


Don't consider if only activated because of ∇
∇ = widening
$\sqcup^\#$ = join
SMT formula "is there a feasible path that makes the candidate invariant grow?"

Abstract Interpretation with PAGAI

[Cousot & Cousot]

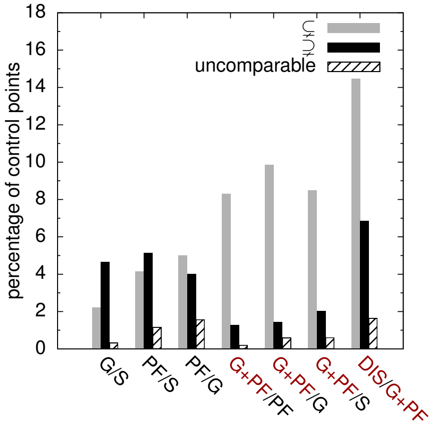

Don't consider if only activated because of ∇
∇ = widening
$\bigsqcup^{\#}$ = join
Keep union symbolic
SMT formula
"is there a feasible path that makes the candidate invariant grow?"

PAGAI: Results

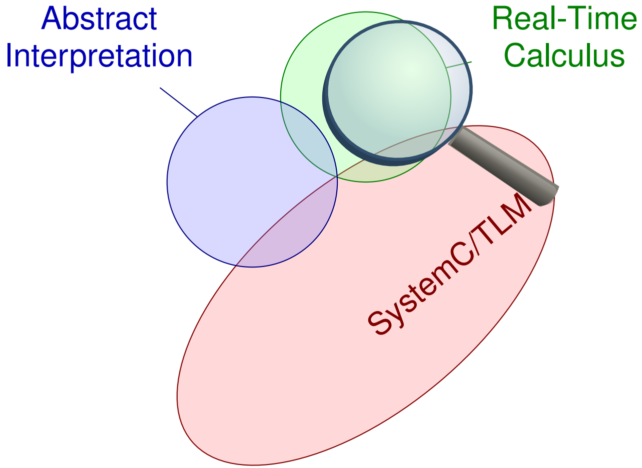
Tested on real programs

Name	kLOC	|*loops*|	Time (in seconds) S	G	PF	C	DIS
a2ps	55	2012	23	74	34	115	162
gawk	59	902	15	46	12	40	50
gnuchess	38	1222	50	220	81	312	351
gnugo	83	2801	77	159	92	766	1493
grep	35	820	41	85	22	65	122
gzip	27	494	22	268	91	303	230
lapack	954	16422	294	3740	3773	8159	10351
make	34	993	67	108	53	109	257
tar	73	1712	37	218	115	253	396

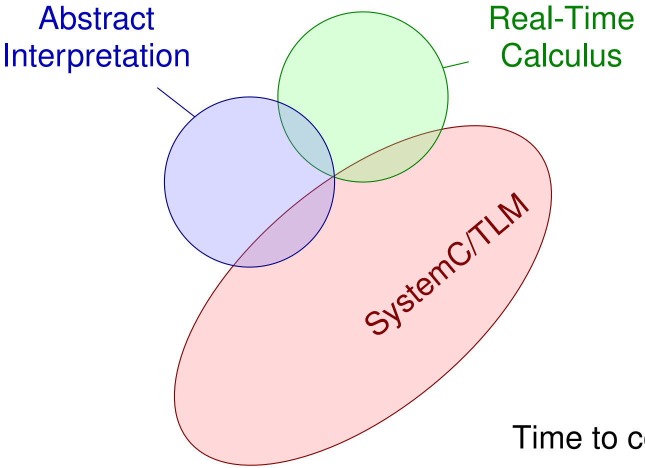
Improves discovered invariants



Outline

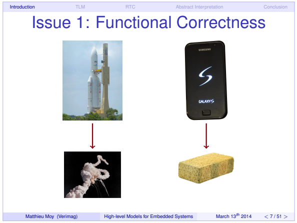


Outline



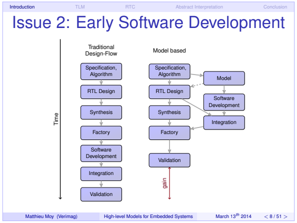
Time to conclude...

Summary


Introduction TLM RTC Abstract Interpretation Conclusion

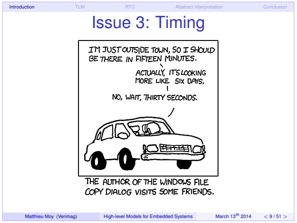
Issue 1: Functional Correctness

Matthieu Moy (Verimag) High-level Models for Embedded Systems March 13th 2014 < 7 / 51 >


Introduction TLM RTC Abstract Interpretation Conclusion

Issue 2: Early Software Development

Matthieu Moy (Verimag) High-level Models for Embedded Systems March 13th 2014 < 8 / 51 >

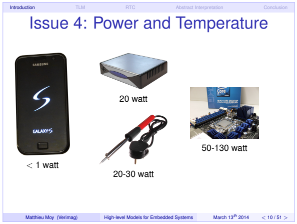

Introduction TLM RTC Abstract Interpretation Conclusion

Issue 3: Timing

I'M JUST OUTSIDE TOWN, SO I SHOULD BE THERE IN FIFTEEN MINUTES.
ACTUALLY, IT'S LOOKING MORE LIKE SIX DAYS.
NO, WAIT, THIRTY SECONDS.

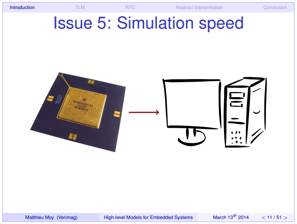
THE AUTHOR OF THE WINDOWS FILE COPY DIALOG VISITS SOME FRIENDS.

Matthieu Moy (Verimag) High-level Models for Embedded Systems March 13th 2014 < 9 / 51 >


Introduction TLM RTC Abstract Interpretation Conclusion

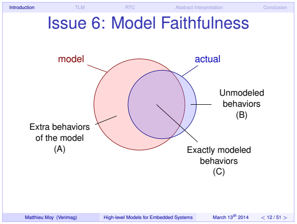
Issue 4: Power and Temperature

Matthieu Moy (Verimag) High-level Models for Embedded Systems March 13th 2014 < 10 / 51 >


Introduction TLM RTC Abstract Interpretation Conclusion

Issue 5: Simulation speed

Matthieu Moy (Verimag) High-level Models for Embedded Systems March 13th 2014 < 11 / 51 >


Introduction TLM RTC Abstract Interpretation Conclusion

Issue 6: Model Faithfulness

model
actual
Unmodeled behaviors (B)
Extra behaviors of the model (A)
Exactly modeled behaviors (C)

Matthieu Moy (Verimag) High-level Models for Embedded Systems March 13th 2014 < 12 / 51 >

Scientific Production

Software Pinapa, PinaVM, PAGAI, LIBTLMPWT, SC-DURING, ac2lus, ...

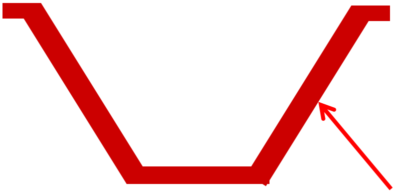
Papers 15 international conferences, 9 workshops, 1 book chapter, 1 journal.

Trained students

- Completed: 1 Ph.D, 5 research master, 6 post-docs, 20 short internships
- Ongoing: 2 Ph.D, 2 research master, 3 short internships

Ensimag course on SystemC/TLM

Condition for Success of a Lab/Industry Cooperation

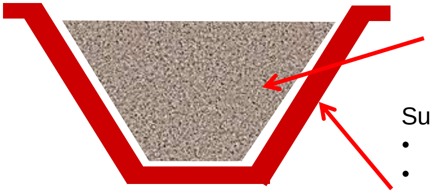


Suitable environment:

- Shared motivation
- Understanding and respect of each party's constraints
- Legal framework
- Bonus: geographic proximity

(Slide from Laurent Maillet-Contoz)

Condition for Success of a Lab/Industry Cooperation

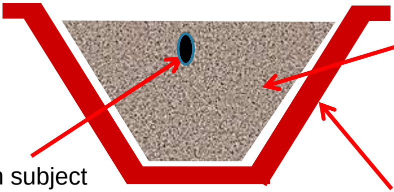


A common ground
- Scientific context
- Applicative context

Suitable environment:
- Shared motivation
- Understanding and respect of each party’s constraints
- Legal framework
- Bonus: geographic proximity

(Slide from Laurent Maillet-Contoz)

Condition for Success of a Lab/Industry Cooperation



A common ground
- Scientific context
- Applicative context

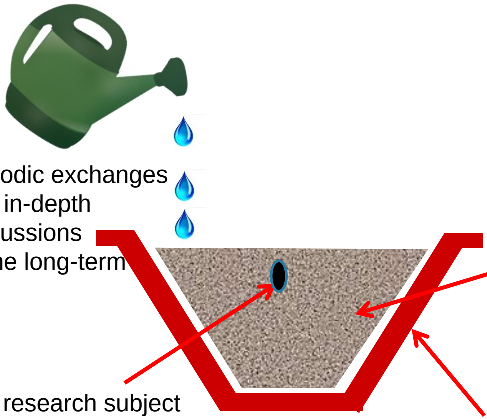
Suitable environment:
- Shared motivation
- Understanding and respect of each party's constraints
- Legal framework
- Bonus: geographic proximity

A research subject covering shared interests

life.augmented

(Slide from Laurent Maillet-Contoz)

Condition for Success of a Lab/Industry Cooperation



Periodic exchanges and in-depth discussions

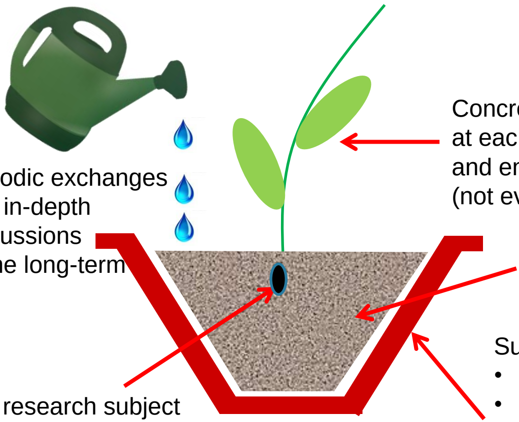
In the long-term

A common ground
- Scientific context
- Applicative context

Suitable environment:
- Shared motivation
- Understanding and respect of each party’s constraints
- Legal framework
- Bonus: geographic proximity

A research subject covering shared interests

Condition for Success of a Lab/Industry Cooperation



Concrete works to progress at each party’s own rythm and enrich the landscape (not everything will succeed)

Periodic exchanges and in-depth discussions
In the long-term

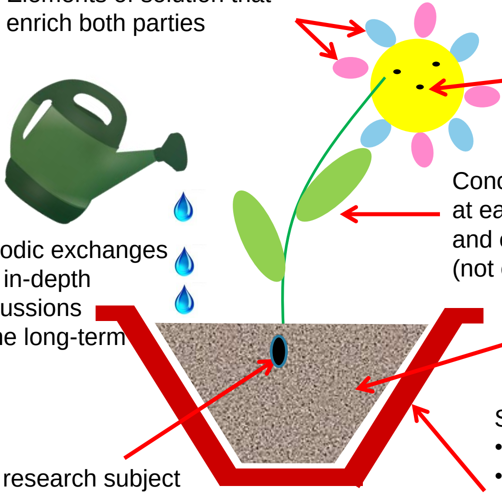
A common ground
- Scientific context
- Applicative context

Suitable environment:
- Shared motivation
- Understanding and respect of each party’s constraints
- Legal framework
- Bonus: geographic proximity

A research subject covering shared interests

(Slide from Laurent Maillet-Contoz)

Condition for Success of a Lab/Industry Cooperation



Elements of solution that enrich both parties

Next cooperation theme

Concrete works to progress at each party's own rythm and enrich the landscape (not everything will succeed)

Periodic exchanges and in-depth discussions In the long-term

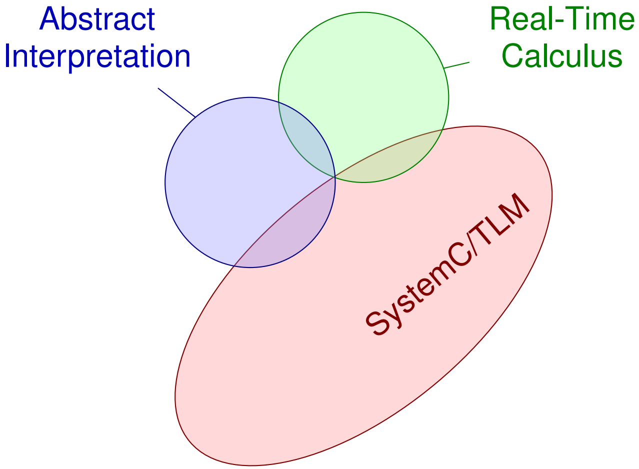
A common ground
- Scientific context
- Applicative context

Suitable environment:
- Shared motivation
- Understanding and respect of each party's constraints
- Legal framework
- Bonus: geographic proximity

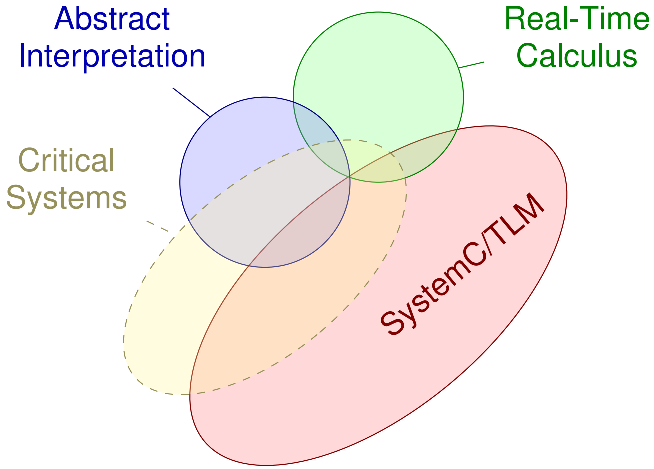
A research subject covering shared interests

(Slide from Laurent Maillet-Contoz)

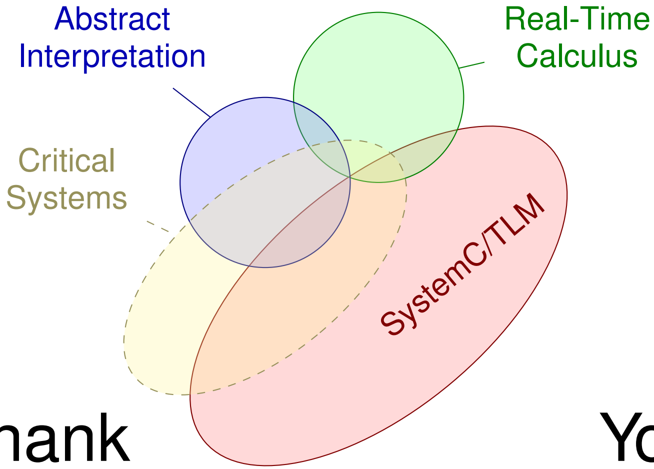
Prospects



Prospects


Abstract Interpretation
Real-Time Calculus
Critical Systems
SystemC/TLM

Questions?

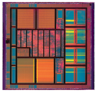


Thank You!

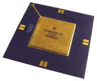
Supervised/Co-supervised Ph.D

Giovanni Funchal	2007 $\rightarrow$ 2011 CIFRE STMicroelectronics co-supervised with Florence Maraninchi
Julien Henry	2011 $\rightarrow$ present co-supervised with David Monniaux
Swadhin Mangaraj	2013 $\rightarrow$ present OpenES european project

Sources

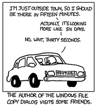


http://en.wikipedia.org/wiki/File:Diopsis.jpg
(Peter John Bishop, CC Attribution-Share Alike 3.0 Unported)



http://www.fotopedia.com/items/flickr-367843750
(oskay@fotopedia, CC Attribution 2.0 Generic)

http://www.flickr.com/photos/artbystevejohnson/4654013143/
(Steve Johnson, CC Attribution 2.0 Generic)

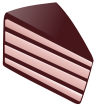

I'M JUST OUTSIDE TOWN, SO I SHOULD BE THERE IN FIFTEEN MINUTES.
ACTUALLY, IT'S LOOKING MORE LIKE SIX DAYS.
NO, WAIT, THIRTY SECONDS.
THE AUTHOR OF THE WINDOWS FILE COPY DIALOG VISITS SOME FRIENDS.

http://xkcd.com/612/
(Randall Munroe, CC Attribution 2.0 Generic)

http://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:Soldering_iron_%28UK_Plug%29.jpg
(oomlout, CC Attribution Share-Alike 2.0 Generic)

Sources

http://commons.wikimedia.org/wiki/File:Intel_Core_I7-920_Boxed_-_15.JPG
(Alan Lorenzo, CC Attribution-Share Alike 3.0 Unported)



http://commons.wikimedia.org/wiki/File:Choc_cake_ill_01.svg
(Kilom691, CC Attribution-Share Alike 3.0 Unported)

http://en.wikipedia.org/wiki/File:Singapore_Airlines_A380_9V-SKH.jpg
(Simon_sees, CC Attribution 2.0 Generic)

http://opensource.org/logo-usage-guidelines
(OSI, Creative Commons Attribution 3.0 License)

+ Open Clip Art public domain pictures