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

Matthieu Moy

Verimag (Grenoble INP)
Grenoble, France

March 13\textsuperscript{th} 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

Teaching at Ensimag

Grenoble INP

The SystemC guy

The RTC guy

Also works on abstract interpretation

Matthieu Moy (Verimag)
About Me

Research at Verimag

Teaching at Ensimag
About Me

The SystemC guy

Research at Verimag

Grenoble INP

Teaching at Ensimag

Matthieu Moy (Verimag)
About Me

The SystemC guy

Research at Verimag

The RTC guy

Teaching at Ensimag
About Me

The SystemC guy

Research at Verimag

Teaching at Ensimag

The RTC guy

Also works on abstract interpretation
About Me

The SystemC guy

Research at Verimag

The RTC guy

Also works on abstract interpretation

Teaching at Ensimag

The macarons guy

Matthieu Moy (Verimag)

High-level Models for Embedded Systems

March 13th 2014
What Are Processors For?

Computers

Source: http://skepchick.org/2013/03/planet-of-the-arthropods/
What Are Processors For?

Computers

Embedded systems

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

Embedded systems ≈ 98%

Vertebrate ≈ 4%

Insects ≈ 96%

Source: http://skepchick.org/2013/03/planet-of-the-arthropods/
Prehistory: My Phone (2010)

```
<table>
<thead>
<tr>
<th>Connectivity</th>
<th>Multimedia Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio IF</td>
<td>12MP Camera IF</td>
</tr>
<tr>
<td>IIS/AC97/PCM</td>
<td>1080 MFC</td>
</tr>
<tr>
<td>S/PDIF</td>
<td>2D VG / 3D Graphics</td>
</tr>
<tr>
<td>Storage IF</td>
<td>NTSC / PAL &amp; HDMI</td>
</tr>
<tr>
<td>HS-MMC/SD</td>
<td>JPEG Codec</td>
</tr>
<tr>
<td>ATA</td>
<td></td>
</tr>
<tr>
<td>ConnectivityManager</td>
<td>16/18/24bpp</td>
</tr>
<tr>
<td>USB Host2.0/OTG2.0</td>
<td>XGA resolution</td>
</tr>
<tr>
<td>UART</td>
<td>3 windows</td>
</tr>
<tr>
<td>IIC</td>
<td>8-bit alpha-blending</td>
</tr>
<tr>
<td>HS-SPI</td>
<td></td>
</tr>
<tr>
<td>GPIO</td>
<td></td>
</tr>
<tr>
<td>Configurable</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

System Peripheral
- RTC
- PLLx4
- Timer with PWM (4-ch)
- Watchdog Timer
- DMA (24-ch)
- Keypad (14x8)
- TS-ADC (12-bit/10-ch)

1GHz Cortex A8
- 32KB/32KB I/D Cache
- 800MHz / 1GHz

512KB L2 Cache
- NEON

96KB RAM
- 64KB ROM

Multi-Layer AHB/AXI Bus
- Crypto Engines
- RP

TFT LCD Controller
- 16/18/24bpp
- XGA resolution
- 3 windows
- 8-bit alpha-blending

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

Matthieu Moy (Verimag)
Another Typical Embedded System: my New Camera

Firmware A

Firmware B
Another Typical Embedded System: my New Camera

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

Software

Hardware
Issue 1: Functional Correctness
Issue 1: Functional Correctness
Issue 1: Functional Correctness
Issue 2: Early Software Development

Traditional Design-Flow

- Specification, Algorithm
- RTL Design
- Synthesis
- Factory
- Software Development
- Integration
- Validation
Issue 2: Early Software Development

Traditional Design-Flow

1. Specification, Algorithm
2. RTL Design
3. Synthesis
4. Factory
5. Software Development
6. Integration
7. Validation

Time

Cost > 1,000,000 $!
Issue 2: Early Software Development
Issue 2: Early Software Development

Traditional Design-Flow

1. Specification, Algorithm
2. RTL Design
3. Synthesis
4. Factory
5. Software Development
6. Integration
7. Validation

Model based

1. Specification, Algorithm
2. RTL Design
3. Synthesis
4. Software Development
Issue 2: Early Software Development

Traditional Design-Flow:
- Specification, Algorithm
- RTL Design
- Synthesis
- Factory
- Software Development
- Integration
- Validation

Model based:
- Specification, Algorithm
- RTL Design
- Synthesis
- Model
- Software Development
Issue 2: Early Software Development

Traditional Design-Flow
1. Specification, Algorithm
2. RTL Design
3. Synthesis
4. Factory
5. Software Development
6. Integration
7. Validation

Model based
1. Specification, Algorithm
2. RTL Design
3. Synthesis
4. Factory
5. Integration
6. Validation
7. Model
8. Software Development
Issue 2: Early Software Development

Traditional Design-Flow:
- Specification, Algorithm
- RTL Design
- Synthesis
- Factory
- Software Development
- Integration
- Validation

Model based:
- Specification, Algorithm
- RTL Design
- Synthesis
- Factory
- Software Development
- Integration
- Validation

Gain
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

20-30 watt

50-130 watt
Issue 4: Power and Temperature

20 watt

20-30 watt

50-130 watt
Issue 4: Power and Temperature

- Smartphones: < 1 watt
- Power tools: 20 watt
- Desktop computers: 50-130 watt
Issue 5: Simulation speed
Issue 6: Model Faithfulness

- Extra behaviors of the model (A)
- Unmodeled behaviors (B)
- Exactly modeled behaviors (C)
Models and Virtual Prototypes

Model/Prototype

Real system
Models and Virtual Prototypes

Model/Prototype \( \rightarrow \) Synthesize \( \rightarrow \) Real system
Models and Virtual Prototypes

Model/Prototype \rightarrow (Synthesize) \rightarrow Real system
Models and Virtual Prototypes

(Synthesize)
Compare
Use as specification
...

Model/Prototype

Real system
Context and Collaborations of my Work

2002
My Ph.D

2005
G. Funchal Ph.D
OpenTLM
FoToVP

2010
HELP
Combest

2014
OpenES
CESyMPA

My Ph.D
G. Funchal Ph.D
OpenTLM
FoToVP
HELP
Combest
OpenES
CESyMPA
Context and Collaborations of my Work
Context and Collaborations of my Work

2002

My Ph.D

2005

OpenTLM

G. Funchal Ph.D

2010

FoToVP

HELP

STMicroelectronics

2014

OpenES

Combest

CESyMPA

CEA-LETI

IRISA

ETHZ

INRIA Rhone Alpes

Docea Power

TIMA
Outline

Abstract Interpretation

Real-Time Calculus

SystemC/TLM
Outline

Abstract Interpretation

Real-Time Calculus

SystemC/TLM
TLM and Software Development

Traditional Design-Flow

- Specification, Algorithm
- RTL Design
- Synthesis
- Factory
- Software Development
- Integration
- Validation

Model based

- Specification, Algorithm
- RTL Design
- Synthesis
- Factory
- Integration
- Validation
- Model
- Software Development
TLM and Software Development

Traditional Design-Flow:
- Specification, Algorithm
- RTL Design
- Synthesis
- Factory
- Software Development
- Integration
- Validation

Model based:
- Specification, Algorithm
- RTL Design
- Synthesis
- Factory
- Integration
- Validation

Gain:
- TLM Model

Time

High-level Models for Embedded Systems
March 13th 2014
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

\[ \approx 1000 \text{ Faster than low-level simulations (RTL)} \]
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

- CPU
  - process = C++ code
- ITC
- VGA
- Timer
- TLM Bus
- Data RAM
- Instruction RAM
- GPIO
Uses of Functional Models

Reference for Hardware Validation

Virtual Prototype for Software Development

SPEC
Uses of Functional Models

Reference for Hardware Validation

Virtual Prototype for Software Development

SPEC

RTL

Matthieu Moy (Verimag)
High-level Models for Embedded Systems
March 13th 2014
Uses of Functional Models

Reference for Hardware Validation

Virtual Prototype for Software Development

$\text{TLM} \Rightarrow \text{RTL}$
Uses of Functional Models

Reference for Hardware Validation

Virtual Prototype for Software Development
Uses of Functional Models

Reference for Hardware Validation

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

Matthieu Moy (Verimag)
Non-Functional Models
Timing, Power consumption, Temperature Estimation

Estimated Time/Power/Temperature

Actual

Unmodified Power/ Temperature-Aware Software

Matthieu Moy (Verimag)
Contributions on TLM

Simulation Speed

Timing

Faithfulness

Power/Temperature

Correctness

Early Software Execution

Optimizing

Compiler

Compilation

Formal Verification

Smart FIFO

Parallelization

jTLM

Memory Models

Functional/extra-functional

Cosimulation

Traffic models
Contributions on TLM

- Optimizing Compiler
- Parallelization
- Smart FIFO
- Memory Models
- jTLM
- Traffic models
- Functional/extra-functional
- Cosimulation
- Faithfulness
- Power/Temperature
- Timing
- Compilation
- Correctness
- Formal Verification
- Simulation Speed

Early Software Execution

Introduction | TLM | RTC | Abstract Interpretation | Conclusion

Matthieu Moy (Verimag) | High-level Models for Embedded Systems | March 13th 2014 | < 22 / 51 >
Contributions on TLM

- Optimizing Compiler
- Parallelization
- Smart FIFO
- Memory Models
- jTLM
- Traffic models
- Functional/extra-functional
- Cosimulation
- Power/Temperature
- Faithfulness
- Formal Verification
- Correctness
- Timing

Early Software Execution

Compilation

Simulation Speed

Introduction

Introduction to TLM RTC Abstract Interpretation Conclusion

Matthieu Moy (Verimag) High-level Models for Embedded Systems March 13th 2014
SystemC: Simple Example

```
int sc_main (int, char **) {
    /* Instantiate 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);
    }
};
```

~ 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

```c++
int sc_main (int, char **) {
    /* Instantiate 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);
}
```

Guillaume Sergent
Kevin Marquet
Matthieu Moy (Verimag)
Challenges and Solutions with SystemC Front-Ends

1. C++ is complex (e.g. clang++ ≈ 400,000 LOC)
   \[\sim\] Write a C++ front-end or reuse one (g++, clang++, EDG, ...)

2. Architecture built at runtime, with C++ code
   \[\sim\] Analyze elaboration phase or execute it

```c++
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;
}
```

```c++
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);
    }
};
```

Guillaume Sergent
Kevin Marquet
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

```c++
int sc_main(int, char **) {
    A a("Alice");
    B b("Bob");
    a.socket.bind(b.socket);
    sc_start();
    return 0;
}
```

```
struct A : sc_module {
    /* Instantiate modules */
    A a("Alice");
    B b("Bob");
    /* Connect them together */
    a.socket.bind(b.socket);
    /* and start simulation */
    sc_start();
    return 0;
}
```

Dynamic Approaches

Static Approaches

High-level Models for Embedded Systems

Matthieu Moy (Verimag)
Contributions on TLM

- Optimizing Compiler
- Parallelization
- Smart FIFO
- Memory Models
- jTLM
- Traffic models
- Functional/extra-functional
- Cosimulation
- Correctness
- Formal Verification
- Timing
- Simulation Speed
- Power/Temperature
- Faithfulness
- Compilation

Matthieu Moy (Verimag)

High-level Models for Embedded Systems

March 13th 2014
Contributions on TLM

- Optimizing Compiler
- Parallelization
- Smart FIFO
- Memory Models
- jTLM
- Formal Verification
- Correctness
- Simulation Speed
- Timing
- Faithfulness
- Power/Temperature
- Traffic models
- Functional/extra-functional
- Cosimulation

Compilation

Early Software Execution

Matthieu Moy (Verimag)
Formal Verification: Encoding Approaches

- SystemC program
- Encoding
  - Property
  - Formal language
  - Existing verifier
  - Yes/No/Maybe

Built-in knowledge of SystemC
- Lesar, Nbac (Lustre)
- SMV
- SPIN (Promela)

Kevin Marquet

Kevin Marquet
Contributions on TLM

- Optimizing Compiler
- Parallelization
- Smart FIFO
- Memory Models
- jTLM
- Traffic models
- Functional/extra-functional
- Cosimulation

Early Software Execution
- Simulation Speed
- Correctness
- Timing
- Compilation
- Formal Verification
- Faithfulness
- Power/Temperature

Nabila Abdessaied
Giovanni Funchal
Matthieu Moy (Verimag)

High-level Models for Embedded Systems
March 13th 2014
Contributions on TLM

- Optimizing Compiler
- Parallelization
- Smart FIFO
- jTLM
- Memory Models
- Formal Verification
- Timing
- Power/Temperature
- Functional/extra-functional
- Correctness
- Functional/extra-functional
- Cosimulation
- Early Software Execution

Nabila Abdessaied
Giovanni Funchal
Matthieu Moy (Verimag)

High-level Models for Embedded Systems
March 13th 2014
Contributions on TLM

- Optimizing Compiler
- Parallelization
- Smart FIFO
- Memory Models
- Faithfulness
- Power/Temperature
- Timing
- Correctness
- Formal Verification
- Nabila Abdessaied
- Giovanni Funchal
- Early Software Execution
Contributions on TLM

- Optimizing Compiler
- Parallelization
- Smart FIFO
- Memory Models
- Formal Verification
- Correctness
- Timing
- Simulation Speed
- Power/Temperature
- Functional/extra-functional
- Traffic models
- Cosimulation
- Early Software Execution
- Compilation
- Optimizing Compiler
- Parallelization
- Smart FIFO
- Memory Models
- Formal Verification
- Correctness
- Timing
- Simulation Speed
- Power/Temperature
- Functional/extra-functional
- Traffic models
- Cosimulation
- Early Software Execution

Nabila Abdessaied
Giovanni Funchal
Matthieu Moy (Verimag)

High-level Models for Embedded Systems

March 13th 2014

< 27 / 51 >
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)

\( \sim \) 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)

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

Our proposal = new constructs:
Desynchronisation (1) / Synchronisation (2)
**SC-DURING: The Idea**

- **SC_THREAD_1** → **OS thread_1**
- **SC_THREAD_2** → **OS thread_2**
- **SC_THREAD_N** → **OS thread_N**

- Unmodified SystemC
- Computations delegated to external threads
- Weak synchronization between SystemC and tasks with duration
Simulated Time in SystemC and SC-DURING

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

Process A:

Process P:

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

h()
i()
j()

SystemC

A ----------- B

SC-DURING

P ----------- Q
```
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:**
```plaintext
// computation
f();
// time taken by f
wait(20, SC_NS);
```

**Process P:**
```plaintext
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:
```c
// computation
f();
// time taken by f
wait(20, SC_NS);
```

Process P:
```c
g();
wait(20, SC_NS);
during(15, SC_NS, h);
```
**SC-DURING: Sketch of Implementation**

```cpp
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
}
```

Diagram:

- A: Thread
- B: Create thread
- C: SystemC executes
- D: Wait for completion
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
}

during(d, f);
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
}

during(d, f);
SC-DURING: Sketch of Implementation

```cpp
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
}
```

A - Thread

B - Create thread

C - Wait(d)

during(d, f);
SC-DURING: Sketch of Implementation

```cpp
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
}
```
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
}

during(d, f);

1. Create thread
2. Wait(d)
3. Join()
SC-DURING: Sketch of Implementation

```c
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
}
```

A: SC-DURING: Sketch of Implementation
B: Create thread
C: SystemC executes
1: Create thread
2: SystemC executes
3: Wait for completion

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

+ thread pooling
+ richer synchronization API
Loosely-Timed Models

Fine-grain Timing

Test machine has $4 \times 12 = 48$ cores
Contributions on TLM

- Optimizing Compiler
- Parallelization
- Simulation Speed
- Memory Models
- Functional/extra-functional
- Power/Temperature
- Formal Verification
- Correctness
- Smart FIFO
- Compilation
- Timing
- Traffic models
- jTLM
- Cosimulation
- Early Software Execution
- Optimizing Compiler
- Parallelization
- Simulation Speed
- Memory Models
- Functional/extra-functional
- Power/Temperature
- Formal Verification
- Correctness
- Smart FIFO
- Compilation
- Timing
- Traffic models
- jTLM
- Cosimulation
- Early Software Execution
Contributions on TLM

- Optimizing Compiler
- Parallelization
- Simulation Speed
- Compilation
- Timing
- Memory Models
- Faithfulness
- Functional/extra-functional
- Cosimulation
- Traffic models
- Power/Temperature
- Smart FIFO
- Formal Verification
- Correctness
- Early Software Execution
- Johan L. B. M. Rutten
- Samuel Jones
- jTLM
- Samir Genaim
- Matthieu Moy (Verimag)

High-level Models for Embedded Systems

March 13th 2014
Contributions on TLM

- Optimizing Compiler
- Parallelization
- Smart FIFO
- Memory Models
- Formal Verification
- Correctness
- Timing
- Functional/extra-functional
- Traffic models
- Power/Temperature
- Cosimulation
- Simulation Speed
- Early Software Execution
- Compilation
- jTLM
SystemC and Extra-Functional Solver Cosimulation

SystemC

Power States

Power and Temperature Solver (ATMI, ACEplorer)

Temperature
SystemC and Extra-Functional Solver Cosimulation

SystemC

Power States

Power and Temperature Solver (ATMI, ACEplorer)

Temperature

Claude Helmstetter

Tayeb Bouhadiba

Matthieu Moy (Verimag)
Contributions on TLM

- Optimizing Compiler
- Parallelization
- Smart FIFO
- Memory Models
- jTLM
- Traffic models
- Functional/extra-functional
- Cosimulation
- Power/Temperature
- Timing
- Correctness
- Formal Verification
- Compilation
- Simulation Speed
- Early Software Execution
- Faithfulness

Matthieu Moy (Verimag)
Contributions on TLM

- Optimizing Compiler
- Parallelization
- Smart FIFO
- Memory Models
- jTLM
- Traffic models
- Functional/extra-functional
- Cosimulation
- Faithfulness
- Power/Temperature
- Timing
- Simulation Speed
- Compilation
- Formal Verification
- Correctness
- Early Software Execution

Matthieu Moy (Verimag)  High-level Models for Embedded Systems  March 13th 2014
Precision Vs Algorithmic Complexity

- Precision
- Algorithmic Complexity

Analytical Models

Real-Time Calculus

Unreachable zone

No states

States

Detailed Computational Models

Model Checking

Simulation

Expressivity

Analyzability

useless zone

Matthieu Moy (Verimag)
Precision Vs Algorithmic Complexity

- Real-Time Calculus
- Analytical Models
- Unreachable zone
- Unreachable zone
- No states
- States
- Detailed Computational Models
- Model Checking
- Simulation

Expressivity vs. Analysability

Matthieu Moy (Verimag)
High-level Models for Embedded Systems
March 13th 2014
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

$\alpha^u(t) / \alpha^l(t) : \text{min/max number of events in any window of duration } t.$
Interfacing with Timed Automata

[Chakraborty et al.]
### Timed Automata Vs
Abstract Interpretation and SMT-Solving

<table>
<thead>
<tr>
<th></th>
<th>Large event counters</th>
<th>Large timing constants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timed automata</td>
<td>![Firework]</td>
<td>![Explosion]</td>
</tr>
<tr>
<td>(Uppaal)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMT solving</td>
<td>![Firework]</td>
<td>![Explosion]</td>
</tr>
<tr>
<td>Abstract Interpretation</td>
<td>![Firework]</td>
<td>![Explosion]</td>
</tr>
</tbody>
</table>

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

\[ \delta t \]

\[ \text{#events} \]

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
The Causality Problem

#events

Deadline
6 pages written

$\delta t$

$\alpha'$
The Causality Problem

One page per day

Deadline
6 pages written

#events

\( \alpha^u \)

\( \alpha^l \)

\( \delta t \)
The Causality Problem

#events

Week-end

One page per day

Deadline

6 pages written

\[ \delta t \]

\[ \alpha^u \]

\[ \alpha^l \]
The Causality Problem

#events

Week-end

One page per day

Deadline
6 pages written

Implicit constraint: maximal procrastination = 2 days
The Causality Problem

Causality closure = compute the implicit constraint automatically
Outline

Abstract Interpretation

Real-Time Calculus

SystemC/TLM
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++;
    }
}
Abstract Interpretation with PAGAI

[Cousot & Cousot]
Abstract Interpretation with PAGAI

[Cousot & Cousot]
Abstract Interpretation with PAGAI

[Cousot & Cousot]

\[ \nabla = \text{widening} \]

\[ \bigoplus = \text{join} \]

SMT formula

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

Don't consider if only activated because of \( \nabla \)

Keep union symbolically
Abstract Interpretation with PAGAI

[Cousot & Cousot]

\[ \Delta = \text{widening} \]
\[ \triangledown \] = join

SMT formula

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

Don't consider if only activated because of \[ \Delta \]

Keep union symbolically
Abstract Interpretation with PAGAI

[Cousot & Cousot]
Abstract Interpretation with PAGAI

[Cousot & Cousot]

\[ \triangleright \] = widening

\[ \# \] = join

SMT formula

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

Don't consider if only activated because of \[ \triangleright \]

Keep union symbolic
Abstract Interpretation with PAGAI

[Cousot & Cousot]

\[ \nabla = \text{widening} \]

\[ \# = \text{join} \]

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

Don't consider if only activated because of \( \nabla \)

Keep union symbolic
Abstract Interpretation with PAGAI

[Cousot & Cousot]

$\nabla = \text{widening}$

$\uplus = \text{join}$

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

Don't consider if only activated because of $\nabla$

Keep union symbolic

Matthieu Moy (Verimag)

High-level Models for Embedded Systems

March 13th 2014
Abstract Interpretation with PAGAI

[Cousot & Cousot]

\[ \square = \text{join} \]

\[ \triangledown = \text{widening} \]

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

Don’t consider if only activated because of \[ \triangledown \]

Keep union symbolic
Abstract Interpretation with PAGAI

[Cousot & Cousot]

∇ = widening

⨆ # = join

SMT formula

“is there a feasible path that makes the candidate invariant grow?”
Abstract Interpretation with PAGAI

[Cousot & Cousot]

\[ \nabla = \text{widening} \]

\[ \# = \text{join} \]

Don’t consider if only activated because of \( \nabla \)

SMT formula

“is there a feasible path that makes the candidate invariant grow?”
Abstract Interpretation with PAGAI

[Cousot & Cousot]

\(\triangledown =\) widening

\(\bigcup \# =\) join

Don’t consider if only activated because of \(\triangledown\)

SMT formula

“is there a feasible path that makes the candidate invariant grow?”

Keep union symbolic
### 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 |
| awk    | 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**

![Graph showing percentage of control points for different methods]
Outline

Abstract Interpretation

Real-Time Calculus

SystemC/TLM
Abstract Interpretation
Real-Time Calculus
SystemC/TLM

Time to conclude...
Summary

Issue 1: Functional Correctness

Issue 2: Early Software Development

Problem 3: Timing

Issue 4: Power and Temperature

Issue 5: Simulation speed

Issue 6: Model Faithfulness
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
Suitable environment:
• Shared motivation
• Understanding and respect of each party’s constraints
• Legal framework
• Bonus: geographic proximity
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

A common ground
- Scientific context
- Applicative context

(Slide from Laurent Maillet-Contoz)
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

A common ground
• Scientific context
• Applicative context

A research subject covering shared interests

(Slide from Laurent Maillet-Contoz)
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

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

A common ground
• Scientific context
• Applicative context

A research subject covering shared interests

(Slide from Laurent Maillet-Contoz)
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

A common ground
- Scientific context
- Applicative context

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

A research subject covering shared interests

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

(Slide from Laurent Maillet-Contoz)
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

Elements of solution that enrich both parties

Next cooperation theme

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

A common ground
- Scientific context
- Applicative context

A research subject covering shared interests

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

(Slide from Laurent Maillet-Contoz)
Prospects

Abstract Interpretation

Real-Time Calculus

SystemC/TLM
Prospects

Abstract Interpretation

Real-Time Calculus

Critical Systems

SystemC/TLM

Thank You!
Questions?

Abstract Interpretation

Real-Time Calculus

Critical Systems

SystemC/TLM

Thank You!

Matthieu Moy  (Verimag)  
High-level Models for Embedded Systems  
March 13th 2014
Supervised/Co-supervised Ph.D

Giovanni Funchal 2007 → 2011
CIFRE STMicroelectronics
co-supervised with Florence Maraninchi

Julien Henry 2011 → present
co-supervised with David Monniaux

Swadhin Mangaraj 2013 → present
OpenES european project
Sources

(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)

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)

(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