About Verimag

Established in January 1993, as a joint research lab between Verilog SA and the partners of the IMAG Institute (CNRS, INPG, UJF).

1993 – 1997

• Industrialization of the Lustre language, through the SCADE environment, currently commercialized by Esterel technologies, and widely used in many application areas, including avionics, critical process control, energy production and transport.

• Transfer of results on verification in testing, integrated in the ObjectGeode environment, for SDL.

1997- present :

• Public research lab, associated with CNRS, INPG, UJF.
• Staff: 30 permanent, 10 under contract, 35 Ph. Students
About Verimag - Scientific Policy

Development relying on long-term, strategic industrial partnerships, with main players in the area (eg: Airbus, France Télécom, STMicroelectronics). These aim to ensure a sufficient level of resources, to support basic research and tight connection to the evolution of practical industrial problems.

International Collaboration, through mobility of researchers (eg: summer visits, short and longer term visitors).

Strong participation in national, European, and international R&D projects

• Verimag plays a regional role through the Minalogic French “Pôle de Compétitivité”.

• Coordination of European research in embedded systems, through the Artist1 Accompanying Measure in FP5, and the Artist2 NoE in FP6.
About Verimag - Teams

Synchronous Languages and Systems (Florence Maraninchi)
- Symbolic verification techniques and abstract interpretation
- Test, simulation, early execution
- Component-based design of SoC
- Model-driven implementation techniques for synchronous systems

Distributed and Complex Systems (Yassine Lakhnech)
- Component-based design of embedded systems (theory, tools)
- Validation of distributed systems (SW and system verification & testing)
- Security (protocols, policy testing, mobile code ver, Certification M&T)
- Adaptive systems (QoS control for multimedia systems)

Timed and Hybrid Systems (Oded Maler)
- Specification, verification, testing and code generation
- Timing and schedulability analysis of large systems
- Property-aware implementation of embedded systems,
Modeling and Validation of Real-Time Systems

Alpine Verification Meeting
EPFL, October 6, 2005

Joseph Sifakis
VERIMAG and ARTIST2 NoE
sifakis@imag.fr
The IF toolset: approach

Modeling and programming languages (SDL, UML, SCADE, Java …)

IF: Intermediate Format, based on a general and powerful semantic model

Optimisation and abstraction

Transition systems

- simulation
- test
- verification1
- verification2
- verification3

state explosion
IF toolset: overall architecture

- IF Static Analyzer
- Objecteering: UML, aml2if
- Rational Rose: RT/UML, OMEGA, uml2if
- ObjectGeode: SDL, sdl2if
- TGV: Test Generation
- model construction
- model checking
- guided simulation
- mincost path extraction
- Test Suites
- SPIDER
- LTS
- CADP
- TReX
- LASH
- RMC
- schedules
IF notation: System description

Processes
- extended timed systems
  (non-determinism, dynamic creation)

Interactions
- asynchronous channels
- shared variables

Data
- predefined data types
  (basic types, arrays, records)
- abstract data types
const N1 = ... ; // constants
type t1 = ... ; // types
signal s2(t1, t2), // signals

// signalroutes
signalroute sr1(1) ... // route attributes from P1 to P3

// processes
process P1(N0)
... // data +
behaviour
endprocess;

... process P3(N3)
... endprocess;

Joseph Sifakis   Alpine Verification Meeting, EPFL, Oct. 6 2005
IF notation: Process description

**Process** = hierarchical, timed systems with actions

```plaintext
process P1(N1);
  fpar ... ;

// types, variables, constants, procedures

state s0 ...;
  ...  // transition t1
endstate;

state s1 #unstable...;
  ...  // transitions t2, t3
endstate;

...  // states s2, s3, s4
endprocess;
```

Joseph Sifakis  Alpine Verification Meeting, EPFL, Oct. 6 2005
IF notation: Process description-transition

\[
\text{transition} = \text{urgency} + \text{trigger} + \text{body}
\]

state \( s_0 \)

\[
\text{urgency} \quad \text{eager} \quad \text{provided} \quad x != 10; \\
\text{when} \quad c_2 \geq 4; \\
\text{input} \quad \text{update}(m); \\
\text{body} \quad \ldots \\
\text{nextstate} \quad s_1; \\
\ldots \\
\text{endstate};
\]

\[
\text{untimed guard} \\
\text{timed guard} \\
\text{signal consumption from the process buffer}
\]

= trigger

statement = data assignment

message emission,

process or signalroute creation or destruction, …
IF notation: interactions - delivery policies

peer

server(0)

client(1)

to one specific instance

unicast

server(0)

client(0)

client(1)

client(2)

to a randomly chosen instance

multicast

server(0)

client(0)

client(1)

client(2)

to all instances
IF notation: timed behavior

- operations on clocks
  - set to value
  - deactivate
  - read the value into a variable

- timed guards
  - comparison of a clock to an integer
  - comparison of a difference of two clocks to an integer

```plaintext
state send;
output sdt(self,m,b) to {receiver}0;
set t:= 10;

nextstate wait_ack;
endstate;

state wait_ack;
input ack(sender,c);
...
when 10 < t < 20 ;
...
endstate;
```
IF notation: dynamic priorities

- **priority order** between process instances $p_1$, $p_2$ (free variables ranging over the active process set)

  [priority_rule_name] : $p_1 < p_2$ if condition($p_1$, $p_2$)

- **semantics**: *only maximal enabled processes can execute*

- **scheduling policies**
  - **fixed priority**: $p_1 < p_2$ if $p_1$ instanceof T and $p_2$ instanceof R
  - **run-to-completion**: $p_1 < p_2$ if $p_2 = \text{manager}(0).\text{running}$
  - **EDF**: $p_1 < p_2$ if Task($p_2$).timer < Task($p_1$).timer ($p_1$)
Core components

syntactic transformation tools:
- static analyser
- code generator

C/C++ code
application specific process code
predefined modules (time, channels, etc.)

IF description
parser writer
IF AST
compiler

interaction model
dynamic scheduling
state space representation

LTS exploration tools
-- debugging
-- model checking
-- test generation

Joseph Sifakis  Alpine Verification Meeting, EPFL, Oct. 6 2005
Core components: exploration platform

active instances

process 1

I₁:P₁ I₂:P₁

process 2

I₁:P₂ I₂:P₂

process j

Iₖ:Pₖ

module

Time

interaction
dynamic scheduling

Succ?

Succ!

execution control

run run step step run step

run create run set, reset

Time module

run run run

step step step

run run run

run run run

run run run

run run run

run run run
Case studies: protocols

SSCOP

Service Specific Connection Oriented Protocol


MASCARA

Mobile Access Scheme based on Contention and Reservation for ATM

case study proposed in **VIRES ESPRIT LTR**


PGM

Pragmatic General Multicast

case study proposed in **ADVANCE IST-1999-29082**
Case studies: asynchronous circuits

**timing analysis**

O. Maler et al. *On timing analysis of combinational circuits*. In *Proceedings of the 1st workshop on formal modeling and analysis of timed systems, FORMATS’03, Marseille, France*.

**functional validation**

D. Borrione et al. *Validation of asynchronous circuit specifications using IF/CADP*. In *Proceedings of IFIP Intl. Conference on VLSI, Darmstadt, Germany*
Case studies: Embedded software

Ariane 5 Flight Program
joint work with EADS Launchers

K9 Rover Executive

Component-based modeling – The BIP framework

Layered component model

Scheduler: dynamic priority rules

Interaction Model: Connectors + Interactions

Composition (incremental description)
Component-based modeling - the BIP framework

Encompass heterogeneity of interaction and execution and rely on a **minimal set of constructs and principles** e.g. interaction models + dynamic priorities

Clear separation between behavior and architecture.

- Architecture is a first class entity
- Correctness-by-construction techniques for deadlock-freedom and liveness, based on sufficient conditions on structure (mainly)

Development of a language independent execution/state exploration platform
Interaction models

**Connectors** are maximal sets of compatible actions

**Interactions** are subsets of connectors; they are defined by using typing \((\text{complete} \downarrow, \text{incomplete} \bullet)\): either they are maximal or they contain some complete interaction

Interactions:
\{tick1,tick2,tick3\}, \{out1\}, \{out1,in2\}, \{out1,in3\}, \{out1,in2, in3\}
Interaction models - examples

- **Models Examples**

  1. **Model 1**: `cl1, cl2`  
     - **CN**: `{cl1, cl2}`  
     - **MI**: `∅`

  2. **Model 2**: `out, in`  
     - **CN**: `{out, in}`  
     - **MI**: `{out}`

  3. **Model 3**: `in1, out, in2`  
     - **CN**: `{in1, out, in}`  
     - **MI**: `{out}`
Interaction models – operational semantics

CN: \{put, get\}, \{prod\}, \{cons\}
MI: \{prod\}, \{cons\}

Operational Semantics

Semantics

Operational
Interaction models - composition

CN[P,C]: {put, get}
MI[P,C]: ∅

CN[P]: {put}, {prod}
MI[P]: {prod}

CN[C]: {get}, {cons}
MI[C]: {cons}

prod  put  get  cons

prod  put

get  cons

prod  put  get  cons

prod  put

get  cons
Priority Systems

Priority system = Behavior + A set of dynamic priority rules

Priority rule | Restricted guard $g_1'$
--- | ---
true $\rightarrow$ a1 $\prec$ a2 | $g_1' = g_1 \land \neg g_2$
C $\rightarrow$ a1 $\prec$ a2 | $g_1' = g_1 \land \neg(C \land g_2)$
Priority Systems - FIFO policy

$t_1 \leq t_2 \rightarrow b_1 \ll b_2$

$t_2 \leq t_1 \rightarrow b_2 \ll b_1$

Graph:

- Sleep 1
  - a1
  - Start t1
- Wait 1
  - b1
- Use 1
  - e1

- Sleep 2
  - a2
  - Start t2
- Wait 2
  - b2
- Use 2
  - e2

$\text{a}_1 \rightarrow b_1 \rightarrow \text{start t1}$

$\text{b}_1 \rightarrow \text{wait 1}$

$\text{e}_1 \rightarrow \text{use 1}$

$\text{a}_2 \rightarrow b_2 \rightarrow \text{start t2}$

$\text{b}_2 \rightarrow \text{wait 2}$

$\text{e}_2 \rightarrow \text{use 2}$

#
Priority Systems - EDF policy

\[ D_1 \cdot t_1 \leq D_2 \cdot t_2 \rightarrow b_2 \prec b_1 \]
\[ D_2 \cdot t_2 \leq D_1 \cdot t_1 \rightarrow b_1 \prec b_2 \]
Priority Systems - Composition of priorities

\( pr2 \neq pr1 \)

\[
\begin{array}{c}
pr2 \\
pr1 \\
\vdots \\
\end{array}
\]

\[
\begin{array}{c}
pr1 \\
pr2 \\
\vdots \\
\end{array}
\]

\[
\begin{array}{c}
a \langle^1 b \\
b \langle^2 c \\
a \langle^1 b \\
\end{array}
\]

\[
\begin{array}{c}
b \langle^2 c \\
c \\
a \langle^1 b \\
\end{array}
\]

\[
\begin{array}{c}
\vdots \\
\end{array}
\]

\[
\begin{array}{c}
\vdots \\
\end{array}
\]
Priority Systems - Composition of priorities

We take:

\[
\begin{align*}
pr1 \\ pr2
\end{align*}
\]

\[
\begin{align*}
\cdots
\end{align*}
\]

\[
\begin{align*}
pr1 \oplus pr2
\end{align*}
\]

\[
\begin{align*}
\cdots
\end{align*}
\]

\[pr1 \oplus pr2\] is the least priority containing \(pr1 \cup pr2\)

Results:

- The operation \(\oplus\) is partial, associative and commutative
- \(pr1(pr2(B)) \neq pr1(pr2(B))\)
- \(pr1 \oplus pr2(B)\) refines \(pr1 \cup pr2(B)\) refines \(pr1(pr2(B))\)
- Priorities preserve deadlock-freedom
Priority Systems: mutual exclusion + FIFO

\[ t_1 \leq t_2 \rightarrow b_1 \prec b_2 \quad t_2 \leq t_1 \rightarrow b_2 \prec b_1 \]

\[ \text{true} \rightarrow b_1 \prec e_2 \quad \text{true} \rightarrow b_2 \prec e_1 \]

- Sleep 1
  - Start \( t_1 \)
  - Wait 1
  - Use 1
- Sleep 2
  - Start \( t_2 \)
  - Wait 2
  - Use 2
Priority systems – mutual exclusion

Mutex on R: \( b_1 \prec f_2 \) \( b_2 \prec \{ f_1, b_1' \} \)

Mutex on \( R' \): \( b_1' \prec \{ f_2, b_2 \} \) \( b_2' \prec f_1 \)

Risk of deadlock: The composition is not a priority order!
The BIP framework - fixed priority preemptive scheduling (1)

\[ b_i \prec b_j , \quad r_i \prec r_j , \quad r_i \prec b_j , \quad b_i \prec r_j \quad (\text{access to the resource – priority preserved by composition}) \]

\[ \{b_i, p_j \} \prec \{f_j, \{r_i, p_j \} \prec f_j , \quad n \geq i > j \geq 1 \quad (\text{non pre-emption by lower pty tasks}) \]

**CN:** \( \{b_i, p_j \} \{r_i, p_j \} \) for \( n \geq i, j \geq 1 \)

**MI:** \( a_i, f_i, b_i \) for \( n \geq i \geq 1 \)
The BIP framework - fixed priority preemptive scheduling (2)

\[ b_i \prec b_j, \ r_i \prec r_j, \ r_i \prec b_j, \ b_i \prec r_j \] (access to the resource – pty inherited by composition)

\[ p_i \prec f_j, \text{ if } w_i \text{ or } e_i' \quad n \geq 1 > j \geq 1 \] (non pre-emption by lower pty tasks)

\[ \{b_i, r_i\} \prec \{f_j, p_j\} \quad n \geq 1, j \geq 1 \] (Mutual exclusion)
The execution platform

Interaction model

Priority

Execution kernel

Platform
The BIP framework – implementation: the kernel

Init

Launch atom’s threads

Loop

Wait all atoms

Execute chosen interaction transfer

Choose among maximal

Select

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal

Filter w.r.t. priorities

Ready

Compute legal interactions

Stable

Notify involved atoms

Choose among maximal

Filter w.r.t. priorities

Execute chosen interaction transfer

Choose among maximal
The execution platform

Component Meta-model

Interaction Meta-model

Dynamic priorities Meta-model

Description in L

C ⇒ a < b

Execution kernel

L

⇒ a < b
Implementation - atomic component: abstract syntax

Component: C
Ports: p1,p2, ...
Data: x,y,z, ....
Access: (p1,{x,y,z}), (p2,{x,u,v}),

Behavior:
state s1
  on p1 provided g1 do f1 to state s1'
  ......................  ...... 
  on pn provided gn do fn to state sn'

state s2
  on ..... 

.......... 

state sn
  on .. ..
Implementation - atomic components

```c
run() {
    Port* p;
    int state = 1;
    while(true) {
        switch(state) {
            case 1: p = sync(a, ga, d, gd);
                if (p == a)
                    fa;  state = 2;
                else
                    fd;  state = 3;
                break;
            case 2: p = sync(b, gb, e, ge);
                ...  // Case 2
            case 3:  ...  // Case 3
        }
    }
}
```
Implementation - connectors and priorities: abstract syntax

**Connector:** BUS={p, p', ... , } complete()

**Behavior:**
- on $\alpha_1$ provided $g_{\alpha_1}$ do $f_{\alpha_1}$
- on $\alpha_2$ provided $g_{\alpha_2}$ do $f_{\alpha_2}$

**Priorities:** PR
- if C1 then {($\alpha_1$, $\alpha_2$), ($\alpha_3$, $\alpha_4$), ... }
- if C2 then {($\alpha$, ...), ($\alpha$, ...), ... }
- if Cn then {($\alpha$, ...), ($\alpha$, ...), ... }
Discussion – related approaches

Vanderbilt’s Approach

Semantic Unit
Meta-model

Composition
Operators

Behavior

Operational
Semantics

ASML

.net

Metropolis

Semantic Domains

Quantity
Managers

Media

Behavior

Operational
Semantics

Platform

PTOLEMY

MoC
(Model of Computation)

Directors
Connectors

Behavior

Operational
Semantics

Platform
Discussion – separation of concerns

Design Space: a system is defined as a point of the 3-dimensional space
Separation of concerns: any combination of coordinates defines a system
Discussion – future work: expressiveness(2)

Example: For given $B$, $IM$ and $PR$ which coordination problems can be solved?

Notion of expressiveness different from existing ones which

- Either completely ignore structure
- or use operators where separation between structure and behavior seems problematic e.g. hiding, restriction