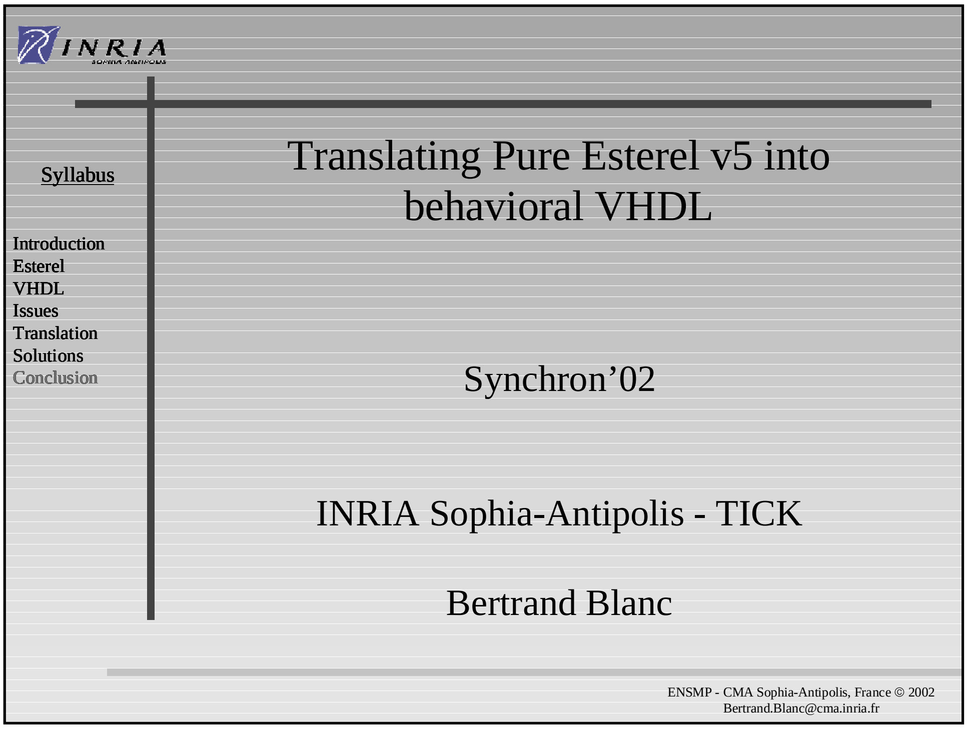


Syllabus

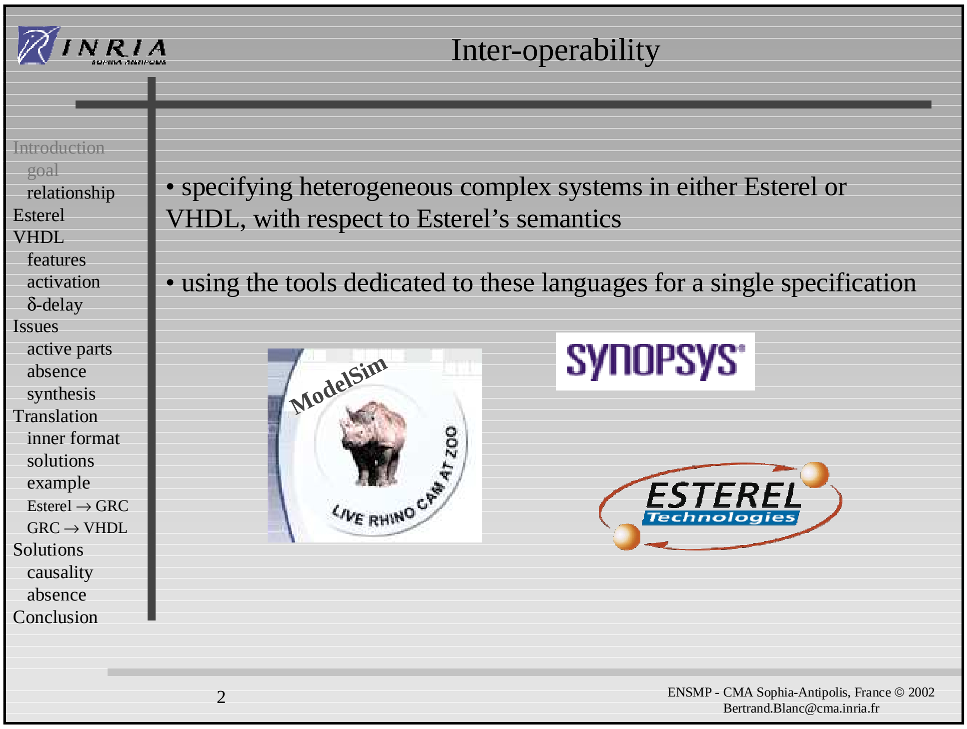
Introduction
Esterel
VHDL
Issues
Translation
Solutions
Conclusion

Translating Pure Esterel v5 into behavioral VHDL

Synchron'02

INRIA Sophia-Antipolis - TICK

Bertrand Blanc



Inter-operability

Introduction
  goal
  relationship
Esterel
VHDL
  features
  activation
  δ-delay
Issues
  active parts
  absence
  synthesis
Translation
  inner format
  solutions
  example
  Esterel → GRC
  GRC → VHDL
Solutions
  causality
  absence
Conclusion

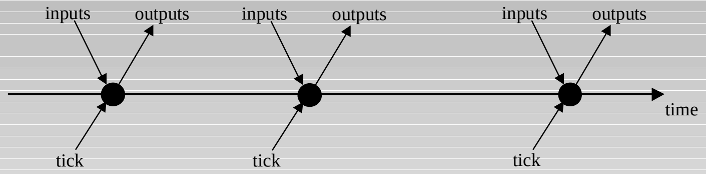
- specifying heterogeneous complex systems in either Esterel or VHDL, with respect to Esterel's semantics

- using the tools dedicated to these languages for a single specification

Relationships Between Esterel & VHDL

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

Study the relationships between Esterel and the high level behavioral VHDL (sequence, loops, concurrent processes and signals)



Esterel has:

- discrete time
- atomic reaction (instant)

We must mimic this model in VHDL

Relationships Between Esterel & VHDL in an Instant

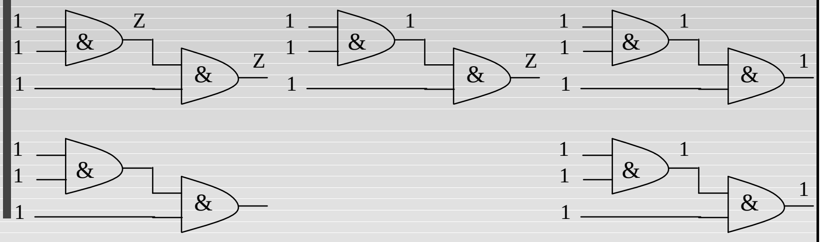
Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

- VHDL (without explicit delays):

the behavior is modeled by the propagation of information through micro-steps. Data should evolve micro-step by micro-step assuming no coherency until stable values get reached.

- Esterel:

Information is propagated with constructive causality ensuring coherent and unique signal statuses.



What about Esterel?

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

Esterel is a synchronous reactive language with instantaneous reaction to absent signal. Its formal semantics unifies activities such as compiling, optimizing and model-checking.

Each equation is computed in each reaction

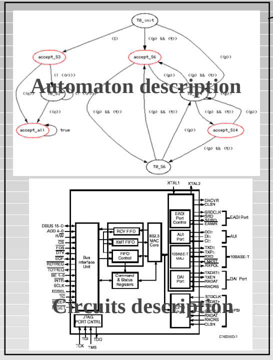
```
E[1]          = E[1] || __ABRO_R[0];
E[3]          = E[3] && !((__ABRO_A1));
__ABRO_R[1]   = E[1] || (__ABRO_R[1] && E[3]);
E[5]          = E[5] && !((__ABRO_A2));
```

compiling into sequential sorted equations, evaluated once per reaction
– acyclic programs:
a variable is to be set before read operations
– cyclic programs:
for constructive programs, an acyclic equivalent form may be generated

VHDL Overview

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

VHDL is a hardware description language devoted to simulating hardware circuits. Its event-driven semantics (signal scheduling) is driven by its simulation engine.

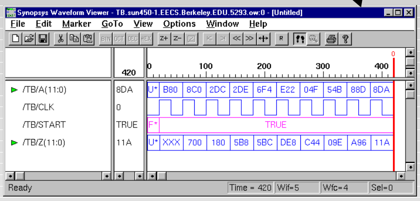


VHDL textual description

compiling

compiler

simulating



```
             420   0      100         200         300         400
/TB/A(11:0)  8DA   U* B80 8C0 2DC 2DE 6F4 E22 04F 54B 88D 8DA
/TB/CLK      0
/TB/START    TRUE  F* TRUE
/TB/Z(11:0)  11A   U* XXX 700 180 5B8 5BC DE8 C44 09E A96 11A
```

Main features of the selected VHDL subset

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

- high level behavioral description = a set of concurrent processes. Each process can be reactivated several times in a same physical time

- processes communicate via signals

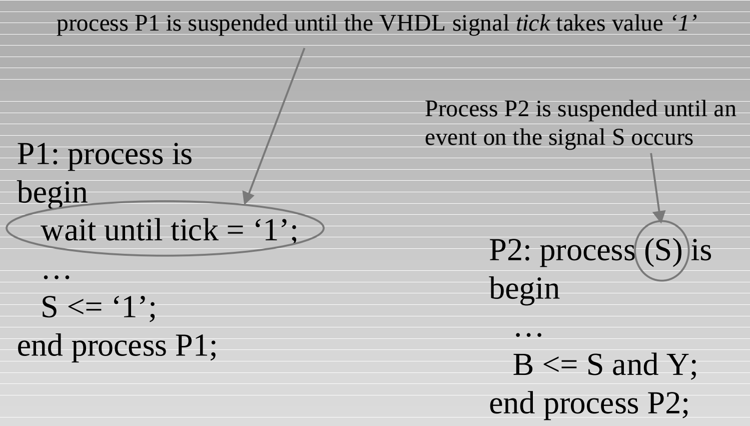
- processes can be activated or suspended
(awaiting to get activated through events on signals)

- signal values are computed through δ-delays (micro-steps)

Each process evolves 0, 1 or several times per reaction

VHDL Example:
Reactivation Condition

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion



process P1 is suspended until the VHDL signal *tick* takes value *‘1’*

Process P2 is suspended until an event on the signal S occurs

```
P1: process is
begin
  wait until tick = ‘1’;
  …
  S <= ‘1’;
end process P1;
```

```
P2: process (S) is
begin
  …
  B <= S and Y;
end process P2;
```

VHDL Example:
δ-Delays

Introduction
- goal
- relationship

Esterel

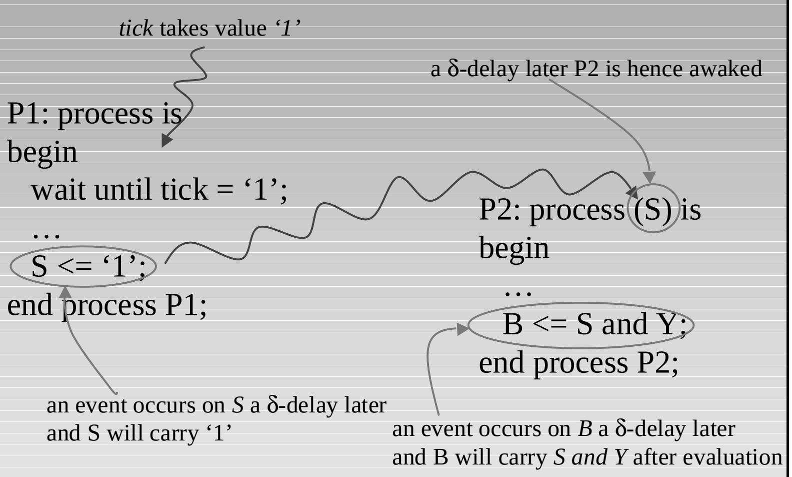
VHDL
- features
- activation
- δ-delay

Issues
- active parts
- absence
- synthesis

Translation
- inner format
- solutions
- example
- Esterel → GRC
- GRC → VHDL

Solutions
- causality
- absence

Conclusion



tick takes value *‘1’*

a δ-delay later P2 is hence awaked

```
P1: process is
begin
  wait until tick = ‘1’;
  …
  S <= ‘1’;
end process P1;
```

```
P2: process (S) is
begin
  …
  B <= S and Y;
end process P2;
```

an event occurs on *S* a δ-delay later
and S will carry ‘1’

an event occurs on *B* a δ-delay later
and B will carry *S and Y* after evaluation

How Run the Only Active Parts?

Introduction
- goal
- relationship

Esterel

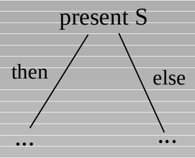
VHDL
- features
- activation
- δ-delay

Issues
- active parts
- absence
- synthesis

Translation
- inner format
- solutions
- example
- Esterel → GRC
- GRC → VHDL

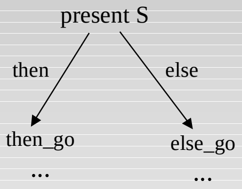
Solutions
- causality
- absence

Conclusion



In the circuit interpretation of Esterel programs, both branches are actually evaluated even if one and only one is active. Either S is present or S is absent!

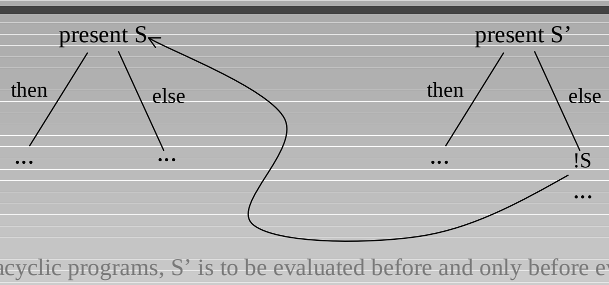
activity seems to be very similar with VHDL simulation
in the case of converging programs



In acyclic programs, when S is present the signal *then_go* is emitted activating the *then* branch. Otherwise, *else_go* is emitted to activate the *else* branch

Reaction to Signal Absence?

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion



In acyclic programs, S' is to be evaluated before and only before evaluating S

Therefore, a part of the topological sorted equations is useful to define reaction to absence.

- how to evaluate only parts which are active?
- how to determine signal absence?
  - – give the control to the *else* branch
  - – evaluate *S'* before *S*

How to do this in VHDL? Processes cannot propagate “non-events”!

Summary of issues

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

- Esterel → VHDL:

  → translating Esterel causality (actions occur before one another) into VHDL δ-delays

  → translating the reaction to absence algorithm into VHDL statements

  → identifying the target subset of VHDL

- *VHDL → Esterel: aims at writing specifications in VHDL with the Esterel's semantics*

INRIA SOPHIA ANTIPOLIS

Chosen Inner Format

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

Dumitru Potop focuses on efficient sequential simulation of Esterel through his inner-format GRC

Keeping on concurrent structure and active parts depicted by an Esterel program

⇒ Dumitru generates sequential code with a static scheduling in order to get efficient simulations

⇒ I translate this inner-format into a VHDL set of concurrent processes instead of sequential VHDL code equivalent to C code

⇒ Avoiding syntactic and lexical issues (parser, inner structure)

INRIA SOPHIA ANTIPOLIS

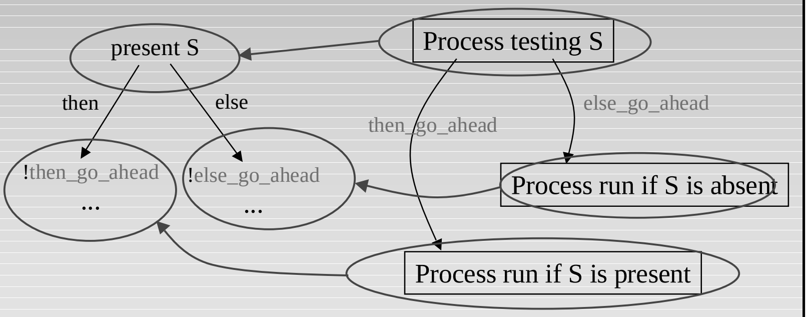
Interpreting GRC Constructions Into VHDL

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

Only what is active must be run

- simulation of acyclic programs expressed as a set of VHDL concurrent processes

- distributed scheduling of acyclic programs (reaction to absence & causality)



A Simple Esterel Program

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

```
module ABO:
  input A, B;
  output O;

loop
  [ await A || await B ] ; emit O
end loop

end module
```

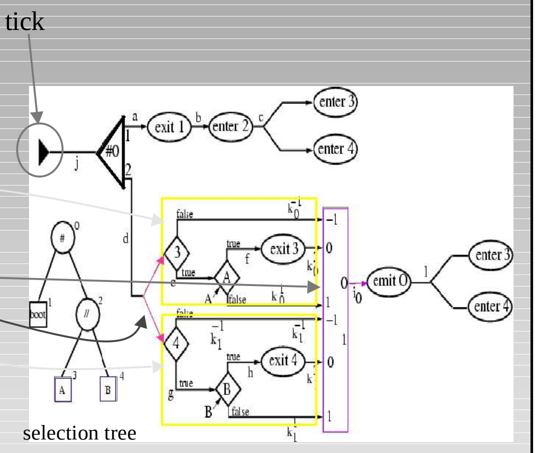
ABO Into GRC

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

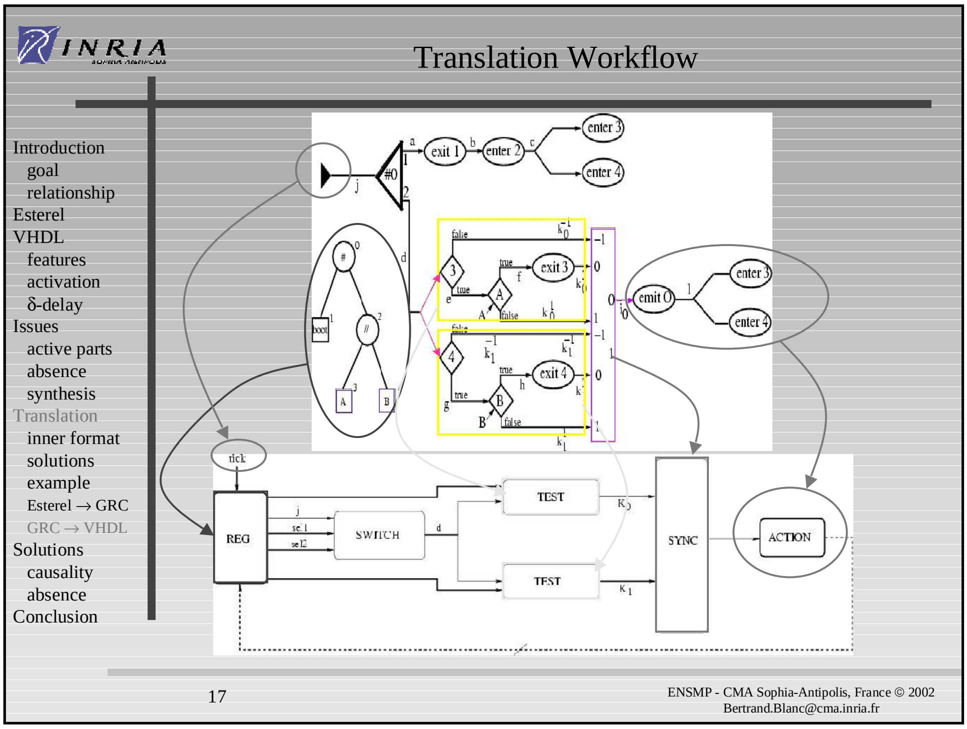
```
module ABO:
  input A, B;
  output O;

loop
  [ await A || await B ] ;
  emit O
end loop

end module
```

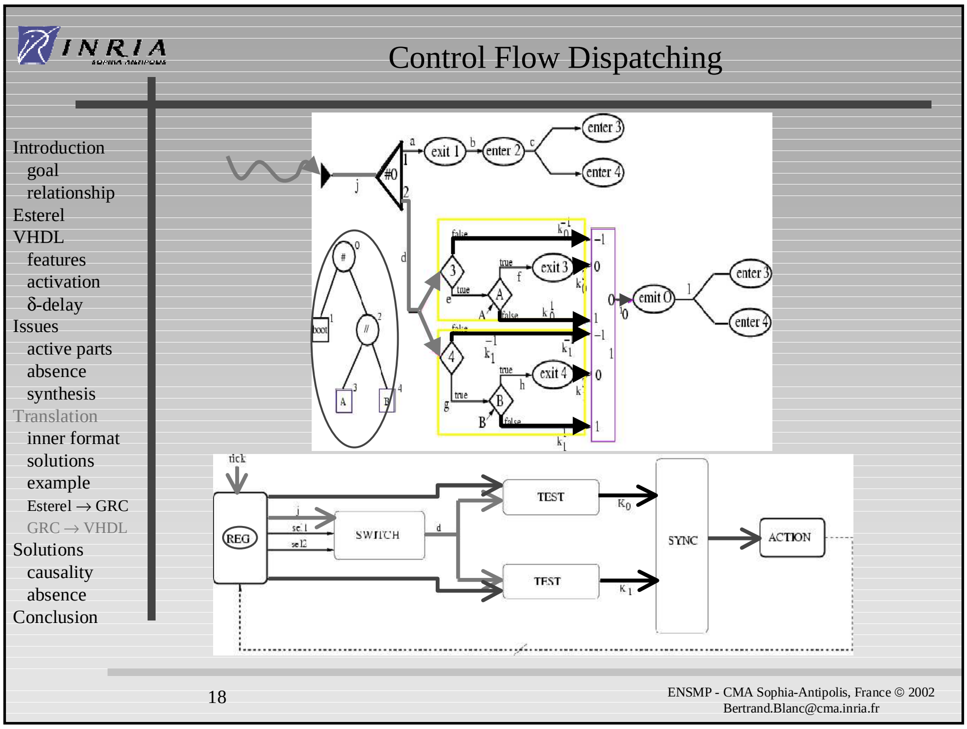


control flow graph



Translation Workflow

Introduction
  goal
  relationship
Esterel
VHDL
  features
  activation
  δ-delay
Issues
  active parts
  absence
  synthesis
Translation
  inner format
  solutions
  example
  Esterel → GRC
  GRC → VHDL
Solutions
  causality
  absence
Conclusion



Control Flow Dispatching

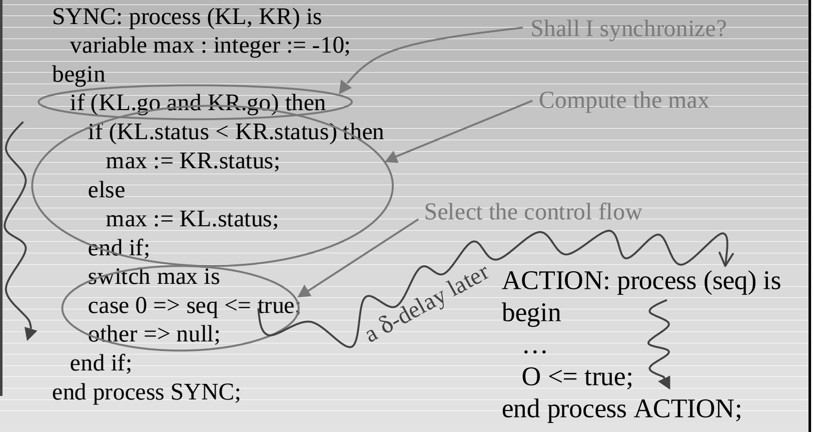
Introduction
  goal
  relationship
Esterel
VHDL
  features
  activation
  δ-delay
Issues
  active parts
  absence
  synthesis
Translation
  inner format
  solutions
  example
  Esterel → GRC
  GRC → VHDL
Solutions
  causality
  absence
Conclusion

Concurrency, Sequence and Causality

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

VHDL signals carry the control flow δ-delay by δ-delay

Example of a 2-signal synchronizer (Esterel → GRC → VHDL):



```
SYNC: process (KL, KR) is
  variable max : integer := -10;
begin
  if (KL.go and KR.go) then
    if (KL.status < KR.status) then
      max := KR.status;
    else
      max := KL.status;
    end if;
    switch max is
    case 0 => seq <= true;
    other => null;
  end if;
end process SYNC;
```

Shall I synchronize?

Compute the max

Select the control flow

a δ-delay later

```
ACTION: process (seq) is
begin
  …
  O <= true;
end process ACTION;
```

Reaction to Absence

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

$\underbrace{\text{present S then <a> end}}_{\text{P1}}$ || $\underbrace{\text{present S' then <b> else emit S end}}_{\text{P2}}$

- S’ present ⇒ nothing to do (at compile-time): at run-time, S’ is emitted in the environment giving the control-flow to P2, hence to <b>.

- S’ absent ⇒ work to be done (at compile-time): at run-time, in acyclic programs, P2 must be activated before P1

→ an extra process that enforces this condition (S’ is set absent) must be generated at compile-time

Reactivate processes through an algorithm computed at compile-time

Algorithm Overview

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

P1 = < present S then <a> end >

P2 = < present S’ then <b> else emit S end >

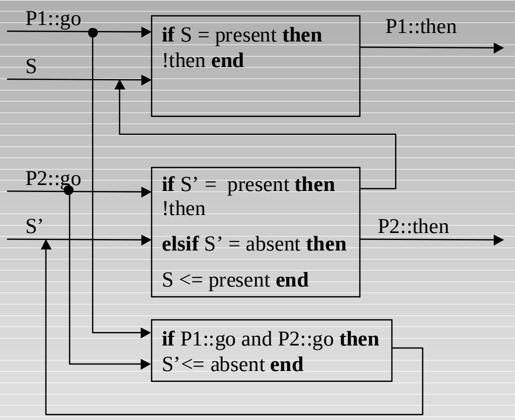
Assumption: S and S’ are local signals (no *emit* elsewhere) → S’ is absent

In VHDL, an add-on process computed at compile-time, wraps the P1 and P2 wires giving control flow, setting S’ absent. The *else* branch in P2 will therefore get enabled through this scheduler, and afterwards P1.

Absence in VHDL

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

In the case where S’ must always be absent



P1::go

S

if S = present **then**
!then **end**

P1::then

P2::go

S’

if S’ = present **then**
!then
elsif S’ = absent **then**
S <= present **end**

P2::then

if P1::go and P2::go **then**
S’<= absent **end**

present S then <a> end || present S’ then <b> else emit S end

Conclusion

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

– Model of computation of pure Esterel v5 into VHDL sticking to Esterel semantics

– Usage of GRC simulation-oriented intermediate format

– Respect causality through activating conditions

– Foundations of reaction to absence through a distributed algorithm

– Generate RTL code from the behavioral VHDL and the sorted equations VHDL:
→ compare the number of wires and registers

Future Work

Introduction
goal
relationship
Esterel
VHDL
features
activation
δ-delay
Issues
active parts
absence
synthesis
Translation
inner format
solutions
example
Esterel → GRC
GRC → VHDL
Solutions
causality
absence
Conclusion

– Generalize the VHDL algorithm implementing reaction to absence to the case of several simultaneous emitters

– Find a better algorithm to encode the states (*selection tree*) used in VHDL
- reduce redundant registers: Dumitru already did it for sequence efficient code.

– Test the scalability of the generated descriptions

– Identify the subset of VHDL and a methodology to implement in VHDL the Esterel semantics

BUT: Attend Ph.D. program at Ecole des Mines de Paris to work on Hybrid Systems.

ECOLE DES MINES DE PARIS

Centre de Mathématiques Appliquées