Synchronous design of embedded systems: the Esterel / Scade approach

Gérard Berry

Chief Scientist

ESTEREL Technologies

www.esterel-technologies.com
Gerard.Berry@esterel-technologies.com

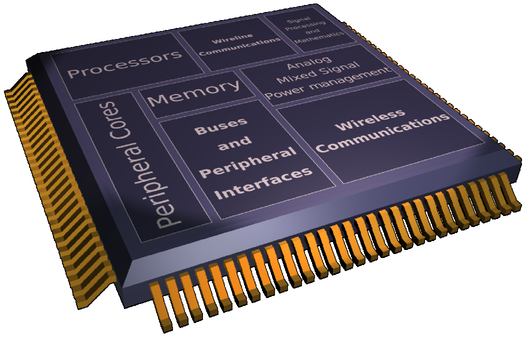
Esterel Technologies - Industries Served

Esterel Studio™

Specification-to-RTL of **hardware IP designs**

- rigorous & unambiguous executable specifications

- automatically-generated efficient RTL / C code



SCADE Suite™

De-facto Standard for **Safety-critical avionics embedded software**

- DO-178B Level A certified systems

- automatically-generated C code

SCADE Drive™

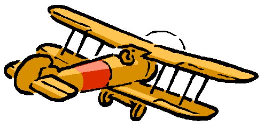
Safety-critical automotive embedded software

- code generator certified by TUV - IEC 61508 standard

Beware of the computer!

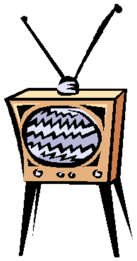
- computers + SoCs = hardware / software mix
- complete change in device interaction
- ever-growing number of critical applications

Applications and Constraints



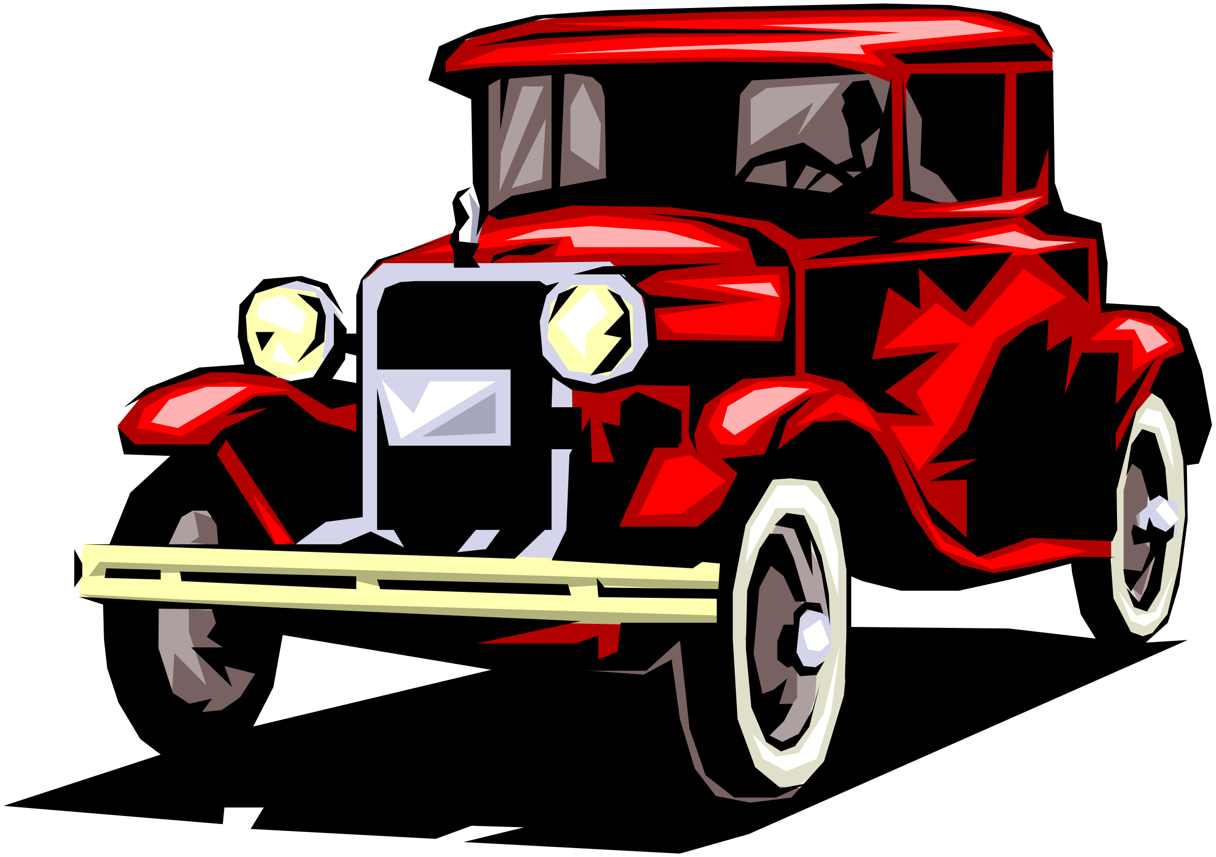
flight-control, engines, brakes, fuel, power, climate
safety-critical => certification

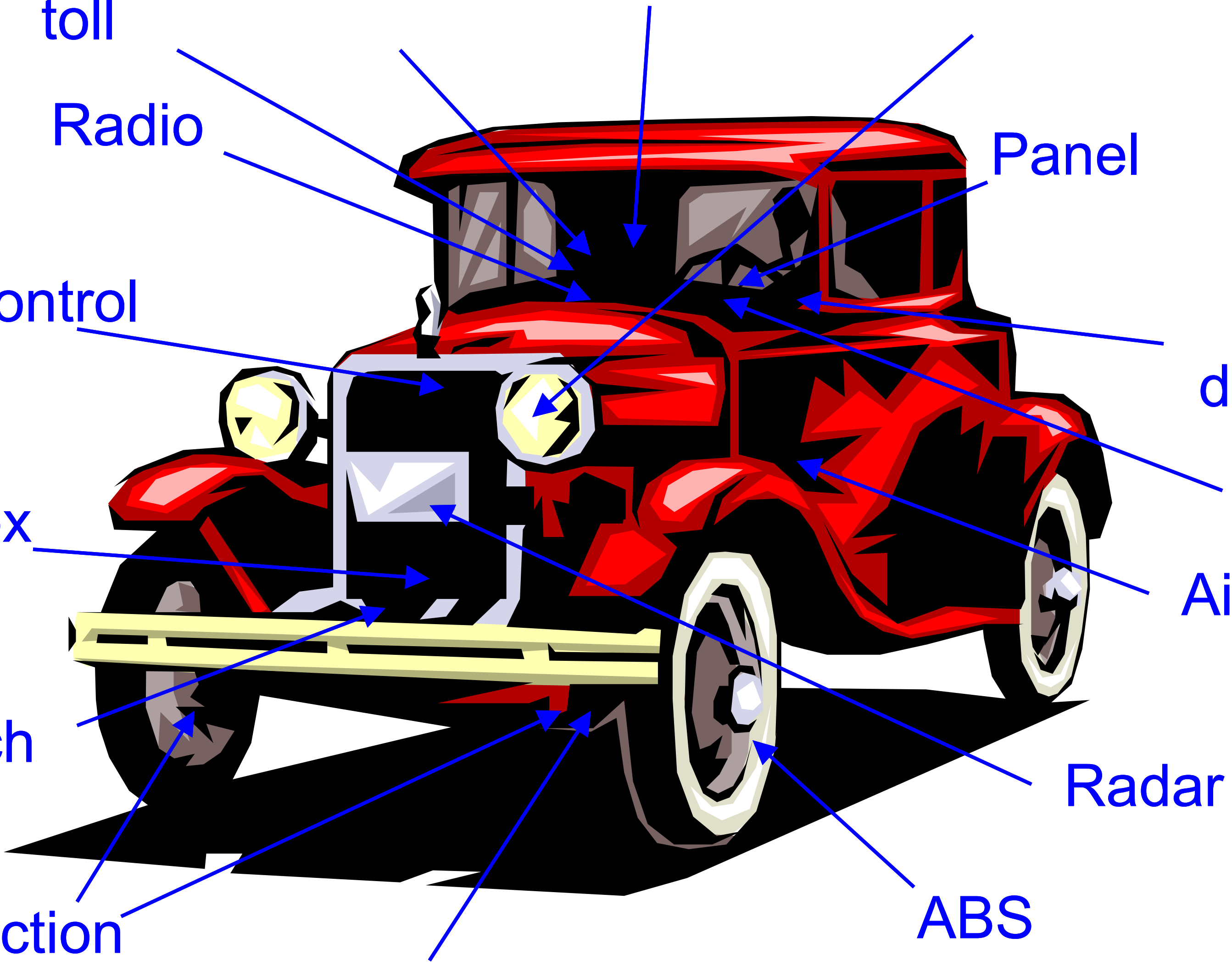
trajectory, attitude, image, telecom
mission-critical => very high quality



telephone, audio, TV, DVD, games
business critical => time-to market + quality

pacemakers, diabet control, robot surgeons
life-critical => TBD (!)





Automatic toll

Air Con

Light control

Alarm detection

Radio

Panel

Engine control

Sleep detector

GPS

Gearbox

Airbags

Clutch

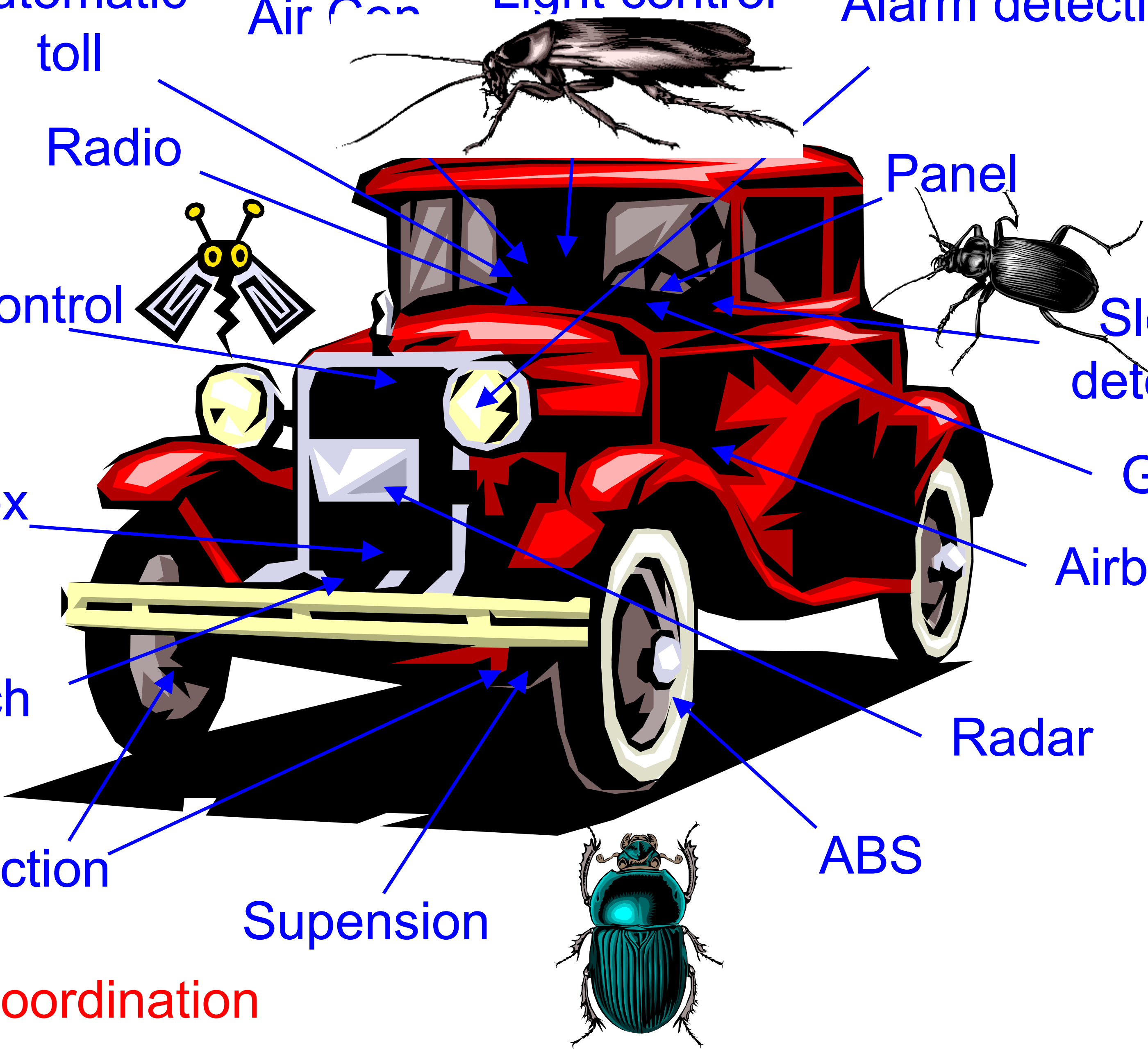
Radar

Direction

ABS

Supension

Global Coordination



Automatic toll

Air Con

Light control

Alarm detection

Radio

Panel

Engine control

Sleep detector

GPS

Gearbox

Airbags

Clutch

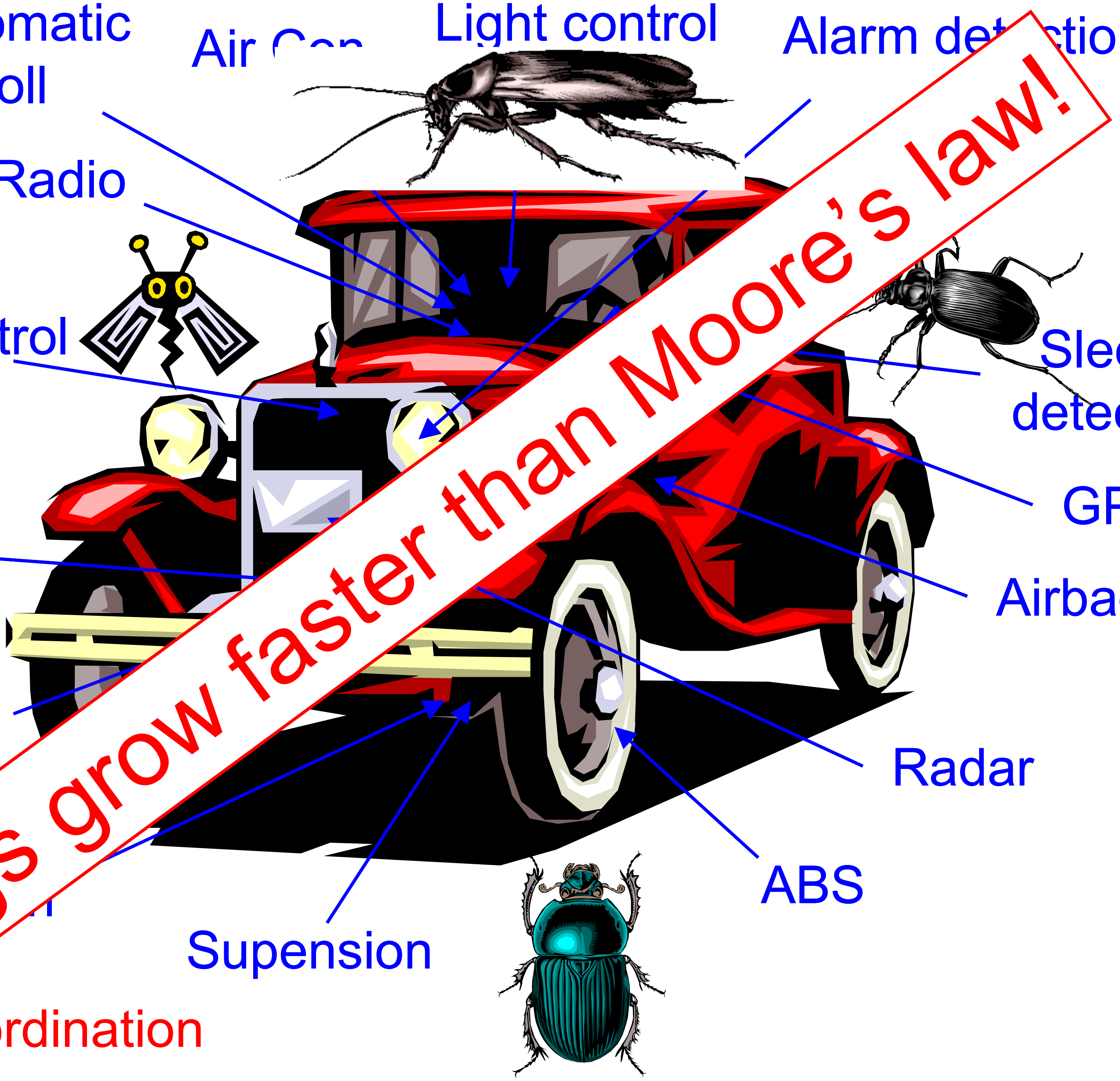
Radar

ABS

Direction

Supension

Global Coordination



How to avoid or control bugs?

- Traditional : better verification by fancier simulation

- Next step : better design

  better and more reusable specifications

  simpler computation models, formalisms, semantics

  reduce architect / designer distance

  reduce hardware / software distance

- Mandatory: better tooling

  synthesis from high-level descriptions

  formal property verification / program equivalence

  certified libraries

- 1982-1985 : first ideas, languages, and semantics

  Esterel (Berry – Rigault, Sophia-Antipolis)

  Lustre (Caspi – Halbwachs, Grenoble)

  Signal (Benveniste – Le Guernic, Rennes)

- 1982-1985 : first ideas, languages, and semantics
  - Esterel (Berry – Rigault, Sophia-Antipolis)
  - Lustre (Caspi – Halbwachs, Grenoble)
  - Signal (Benveniste – Le Guernic, Rennes)
- 1985-1998 : more languages, semantics, compiling & verification
  - SyncCharts (André), Reactive C (Boussinot), TCC (Saraswat)
  - causality analysis (Gonthier, Shiple)
  - links to dataflow (Ptolemy), to hardware (Vuillemin), etc.
  - formal optimization & verification techniques (Madre & Coudert, Touati)

- 1982-1985 : first ideas, languages, and semantics

  Esterel (Berry – Rigault, Sophia-Antipolis)
  Lustre (Caspi – Halbwachs, Grenoble)
  Signal (Benveniste – Le Guernic, Rennes)

- 1985-1998 : more languages, semantics, compiling & verification

  SyncCharts (André), Reactive C (Boussinot), TCC (Saraswat)
  causality analysis (Gonthier, Shiple)
  links to dataflow (Ptolemy), to hardware (Vuillemin), etc.
  formal optimization & verification techniques (Madre & Coudert, Touati)

- 1998 –2001 : maturation, industrial experimentation

  active international research (Edwards, Schneider, Ramesh, etc.)
  applications: avionics, nuclear plant safety, telecom, robotics

- 1982-1985 : first ideas, languages, and semantics

  Esterel (Berry – Rigault, Sophia-Antipolis)
  Lustre (Caspi – Halbwachs, Grenoble)
  Signal (Benveniste – Le Guernic, Rennes)

- 1985-1998 : more languages, semantics, compiling & verification

  SyncCharts (André), Reactive C (Boussinot), TCC (Saraswat)
  causality analysis (Gonthier, Shiple)
  links to dataflow (Ptolemy), to hardware (Vuillemin), etc.
  formal optimization & verification techniques (Madre & Coudert, Touati)

- 1998 –2001 : maturation, industrial experimentation

  active international research (Edwards, Schneider, Ramesh, etc.)
  applications: avionics, nuclear plant safety, telecom, robotics

- 2001-2006 : industrial expansion

  major standard in avionics, expanding in rail, automotive, etc.
  hardware circuit design

Embedded Modules Anatomy

- CC : continuous control, signal processing

  differential equations, digital filtering

  specs and simulation with Matlab / Scilab

- FSM : finite state machines (automata)

  discrete control, protocols, security, displays, etc.

  flat or hierarchical FSMs

- Calc : heavy calculations

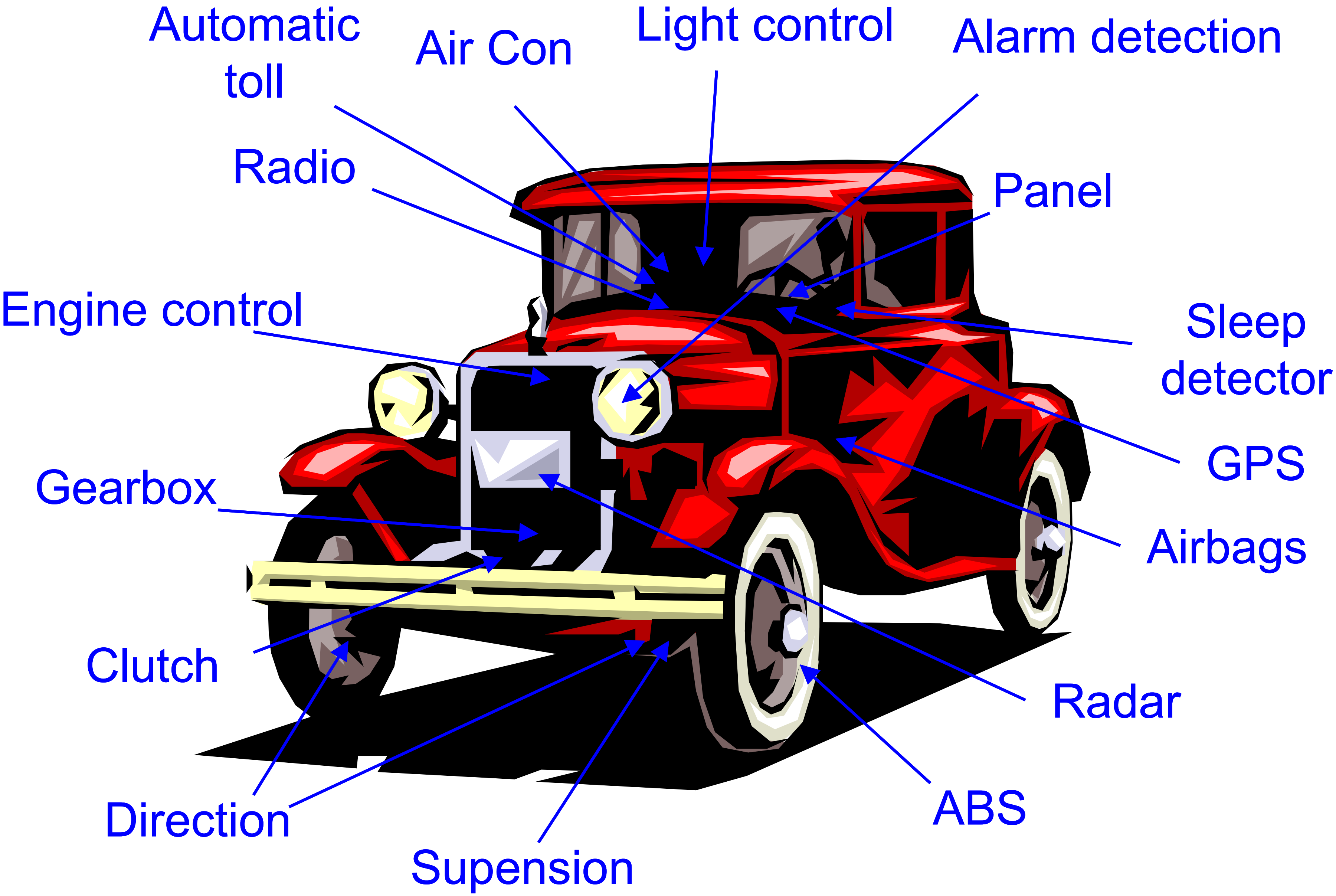
  navigation, encryption, image processing

  C + libraries

- Web : HMI, audio / video

  user interaction / audio / vidéo

  data flow networks, Java



Automatic toll

Air Con

Light control

Alarm detection

Radio

Panel

Engine control

Sleep detector

GPS

Gearbox

Airbags

Clutch

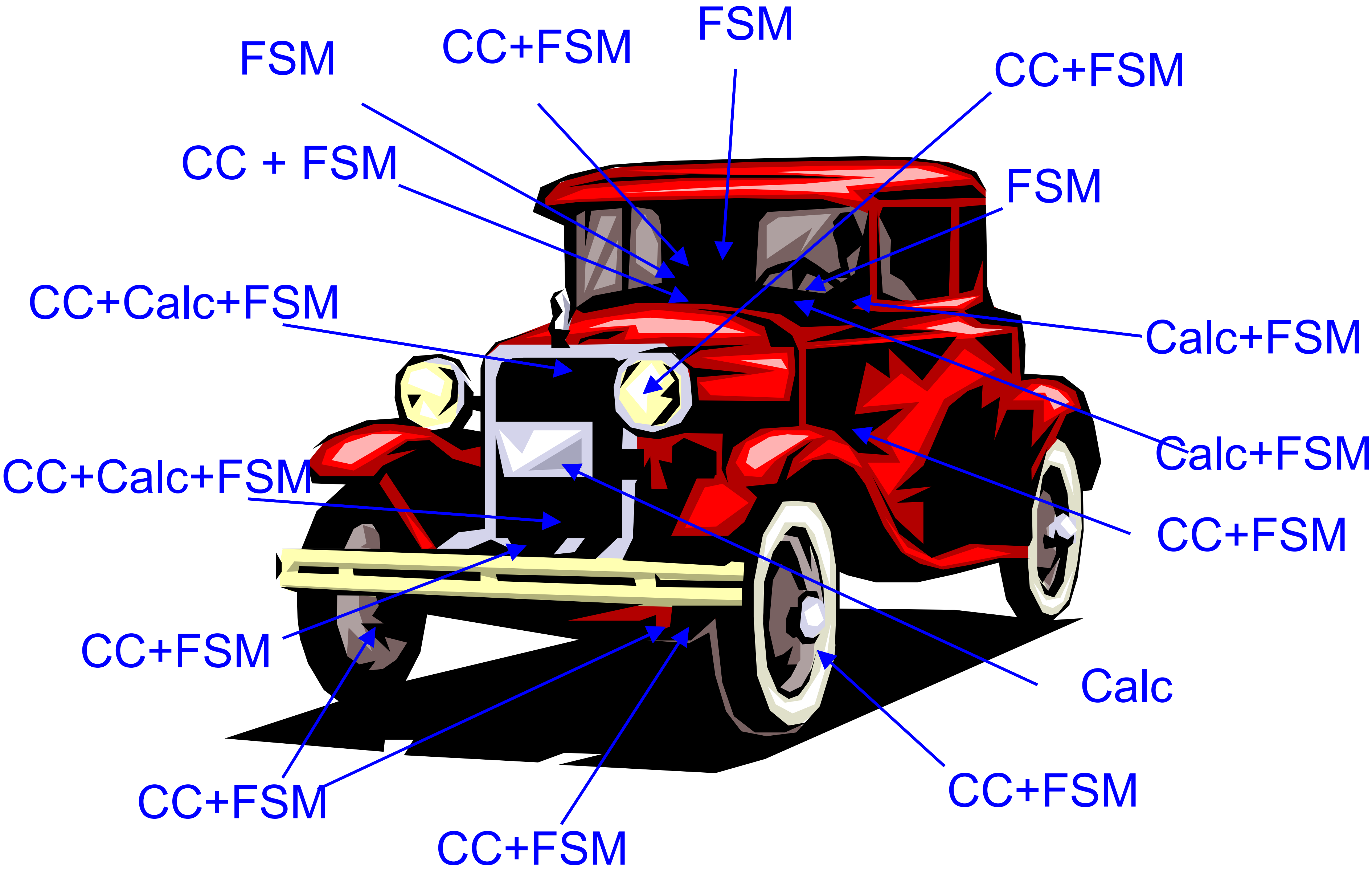
Radar

ABS

Direction

Supension

Global Coordination



FSM

CC+FSM

FSM

CC+FSM

CC + FSM

FSM

CC+Calc+FSM

Calc+FSM

Calc+FSM

CC+Calc+FSM

CC+FSM

CC+FSM

Calc

CC+FSM

CC+FSM

CC+FSM

Global Coordination : Calc+CC+FSM

Key Computation Principles

- Concurrency is fundamental

  implicit in CC, audio / video, protocols, etc.

  also mandatory for Web and Calc

- Determinism is fundamental

  implicit for CC and FSM

  who would drive a non-deterministic car?

  can be relaxed for Web, infotainment, etc.

- Physical distribution becomes fundamental

  separation of functions, links between them

  redundancy for fault-tolerance

  global time needed for distributed control

The Classical Software Development Model is Inadequate

- Turing complete => too rich, too hard to check
- OS- or thread-based concurrency => too hard too check

  interference, non-determinism
- CC implementation too indirect (manual action scheduling)
- Inadapted to circuit design (except for filters)

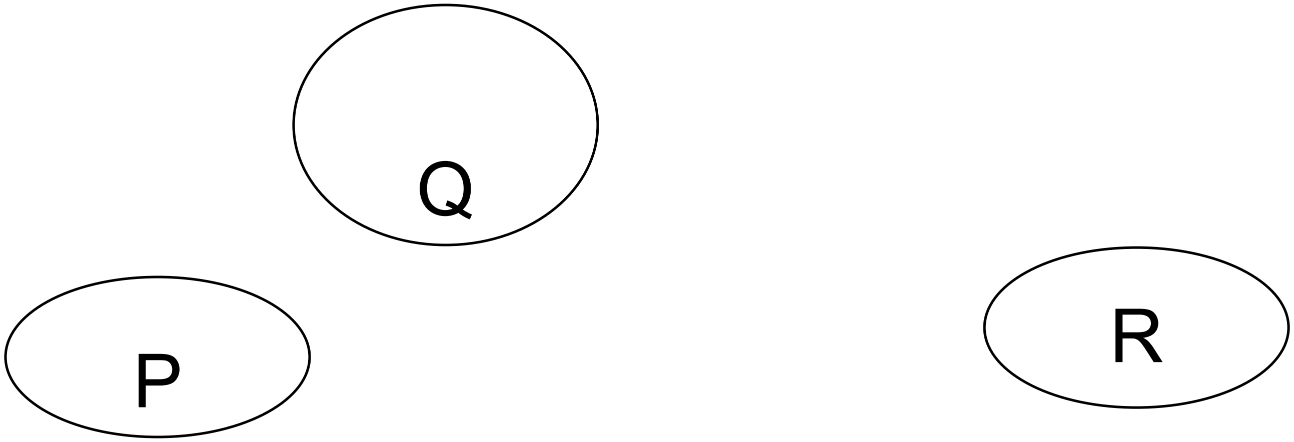
The Classical Hardware Development Model becomes Inadequate

- Structural RTL descriptions hide behavior dynamics
- HDLs inadequate for software
- Concurrency OK, but sequencing very indirect
- Quite old language basis, semantics too vague

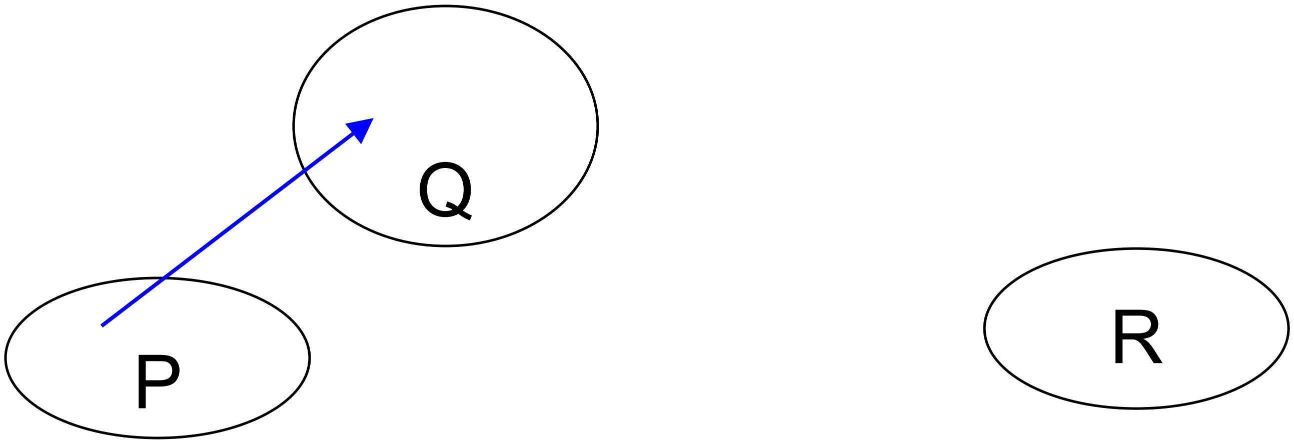
The Classical Hardware Development Model becomes Inadequate

- Structural RTL descriptions hide behavior dynamics
- HDLs inadequate for software
- Concurrency OK, but sequencing very indirect
- Quite old language basis, semantics too vague

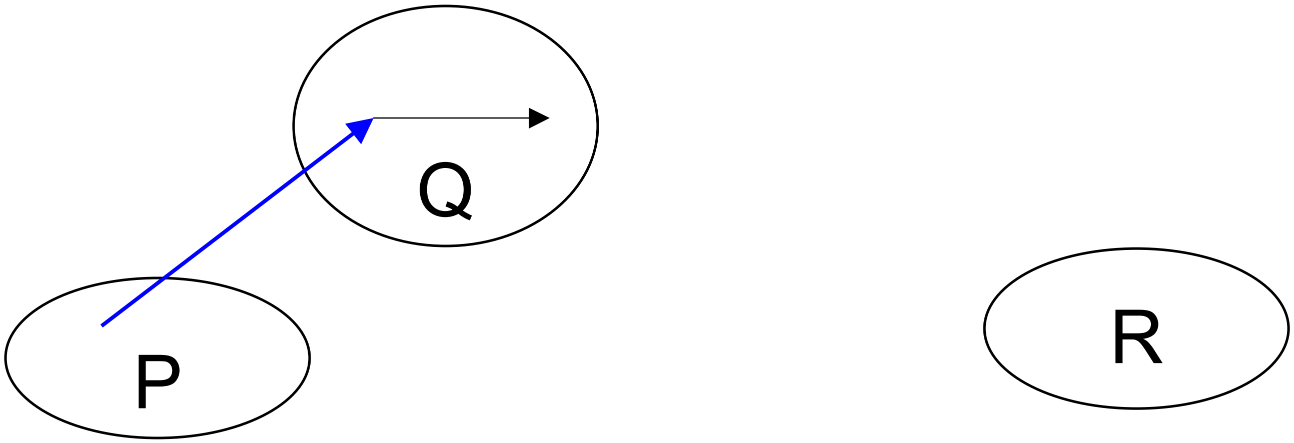
⇒ much simpler models are needed
that reconcile sequencing and concurrency



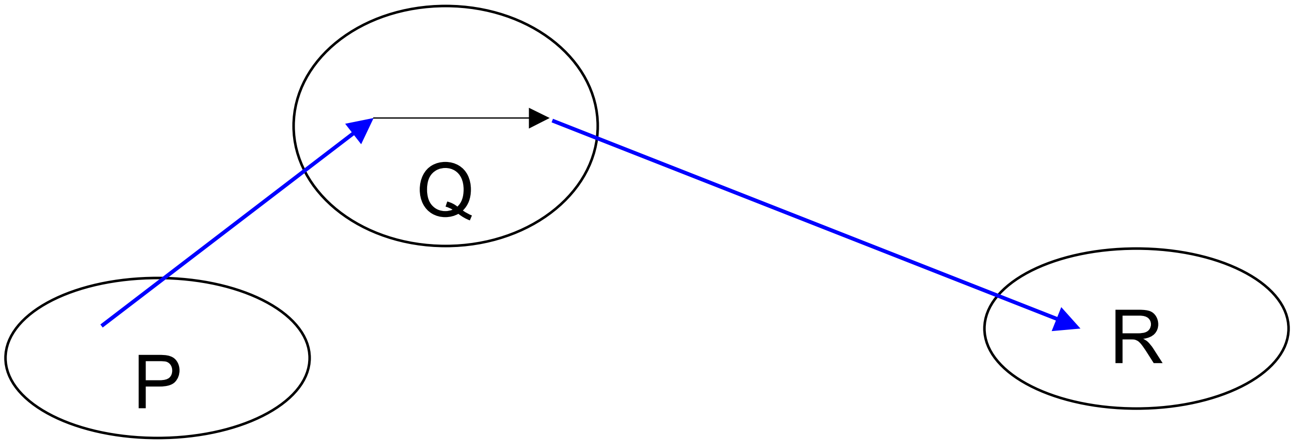
Concurrency : the compositionality principle



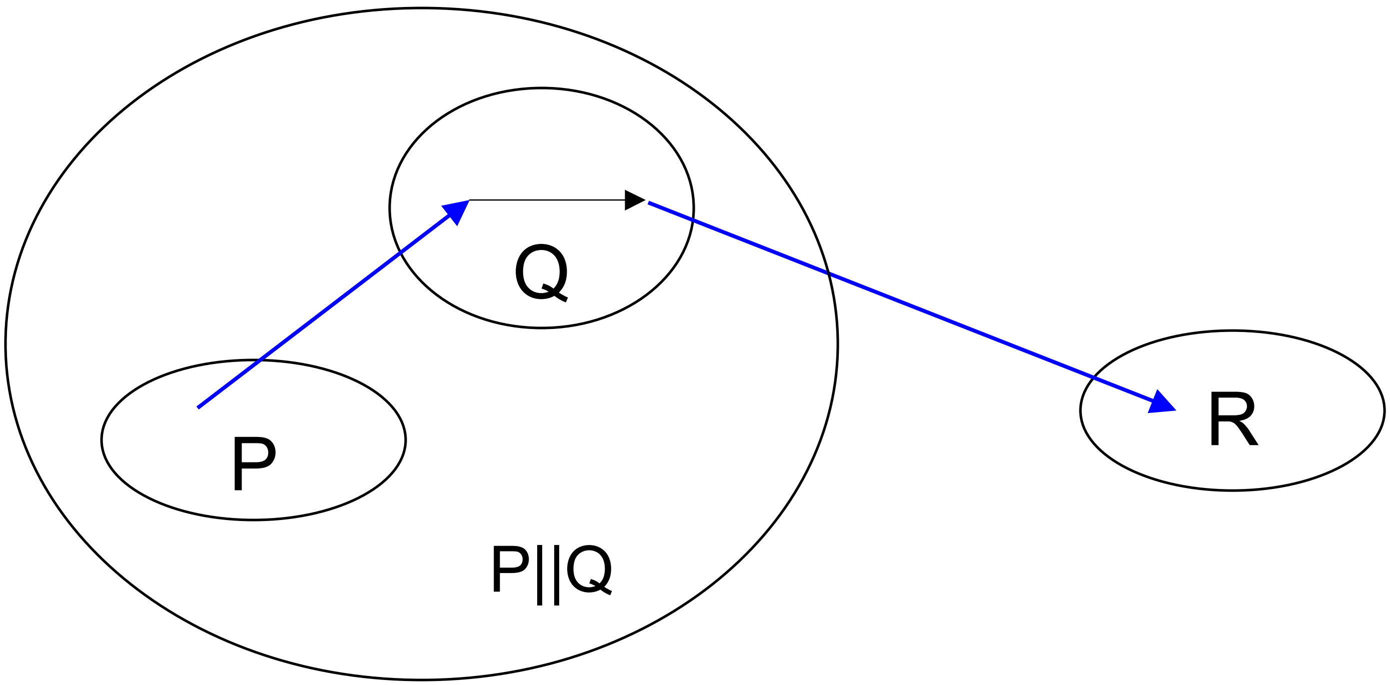
Concurrency : the compositionality principle



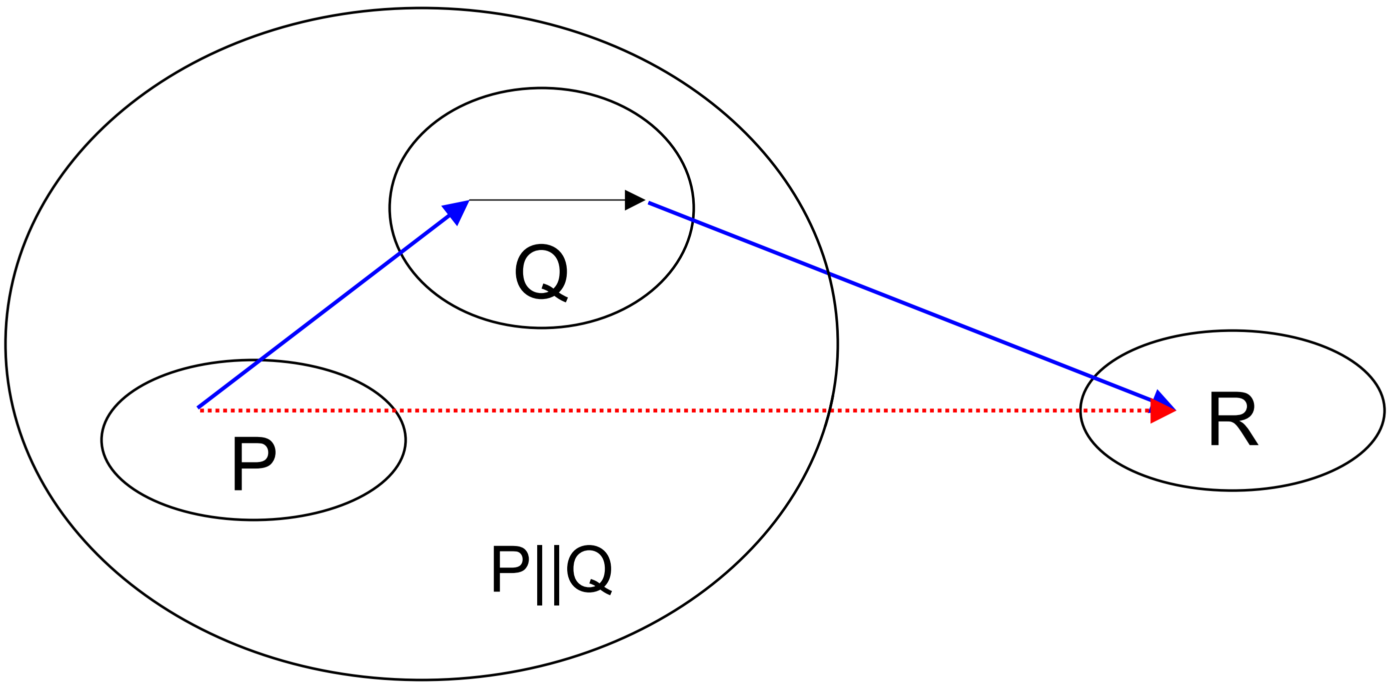
Concurrency : the compositionality principle



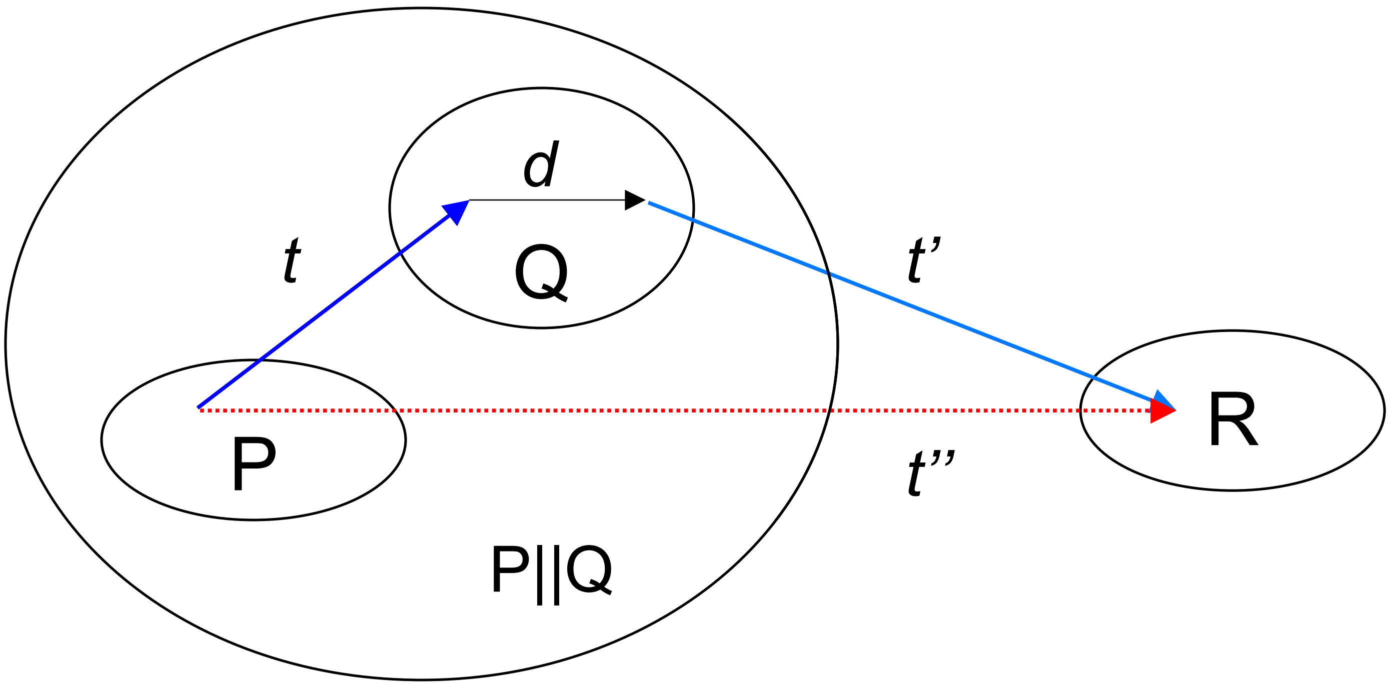
Concurrency : the compositionality principle

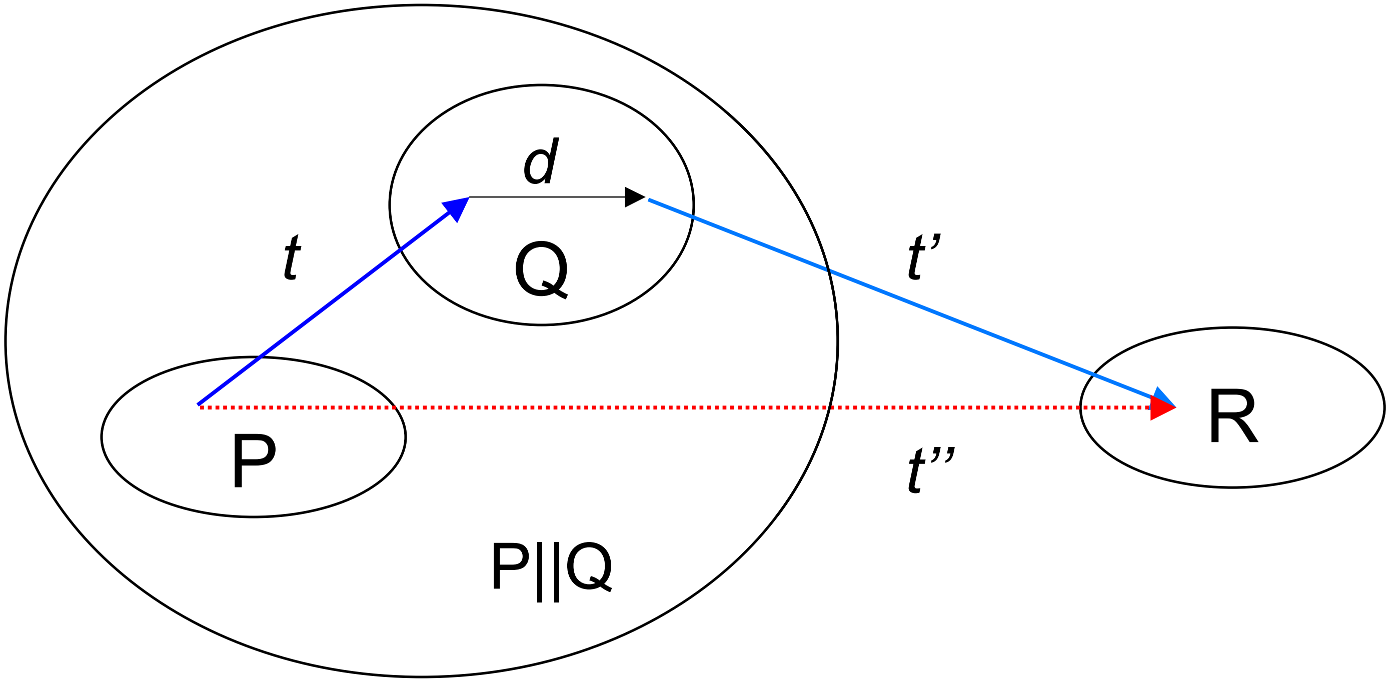


Concurrency : the compositionality principle

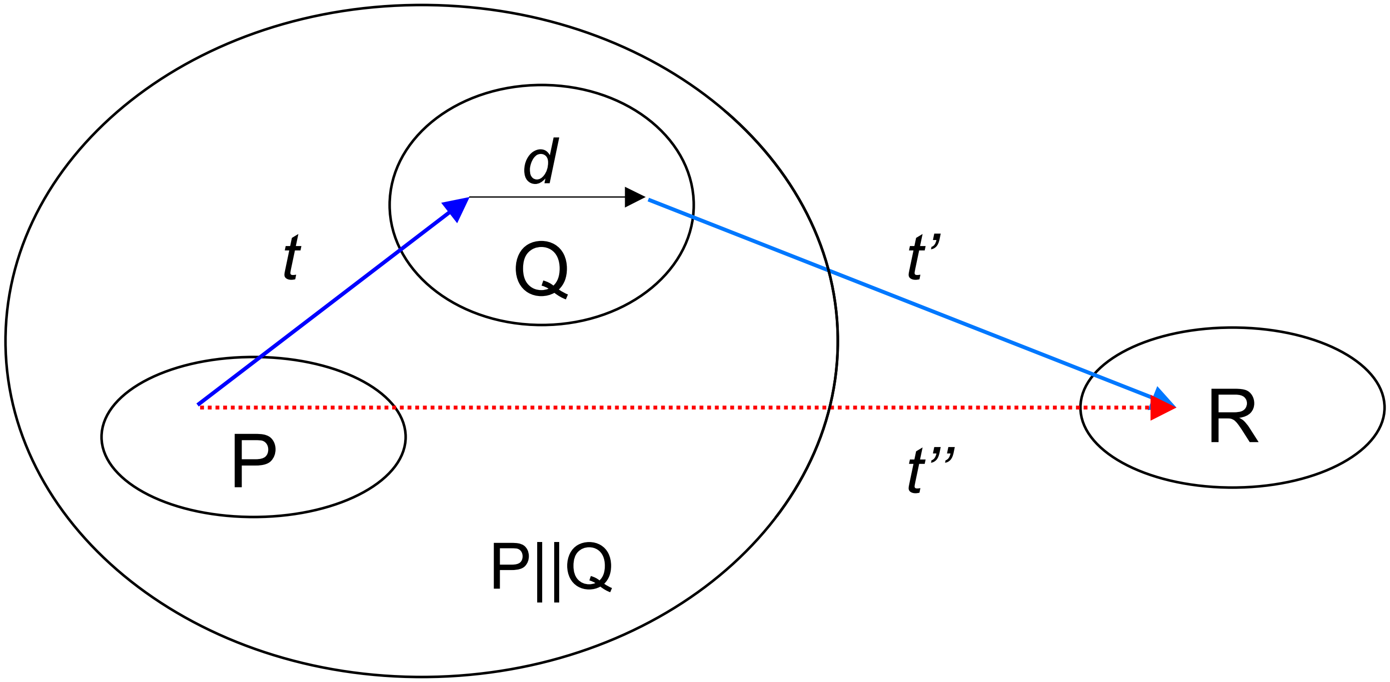


Concurrency : the compositionality principle


d
t
Q
t’
R
P
t’’
P||Q

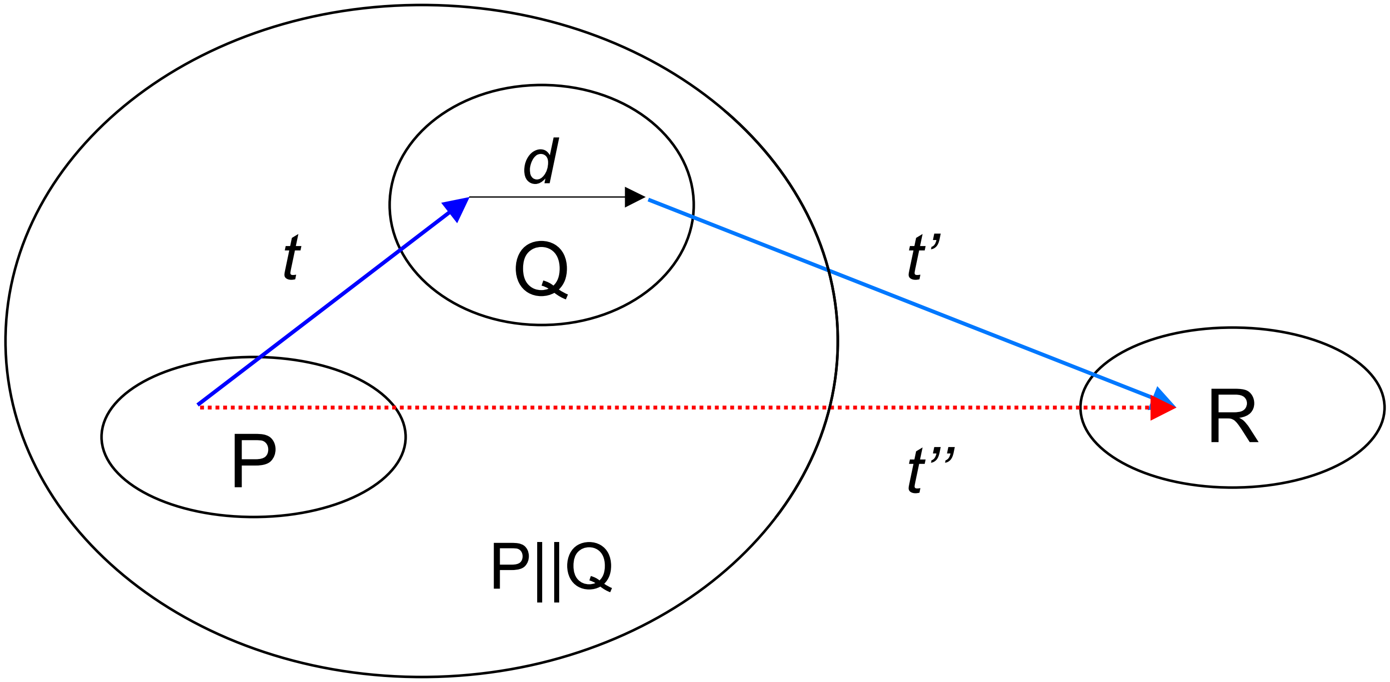


$$t'' = t + d + t'$$



$$t'' = t + d + t'$$

$$t'' \sim t \sim d \sim t'$$



$$t'' = t + d + t'$$

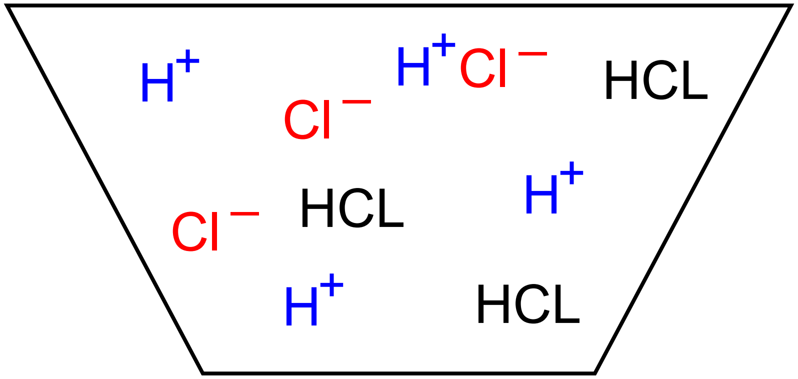
$$t'' \sim t \sim d \sim t'$$

$$t \sim t + t$$

Only 3 solutions :

• *t* arbitrary	asynchrony
• *t* = 0	synchrony
• t predictable	vibration

Arbitrary Delay : Brownian Motion



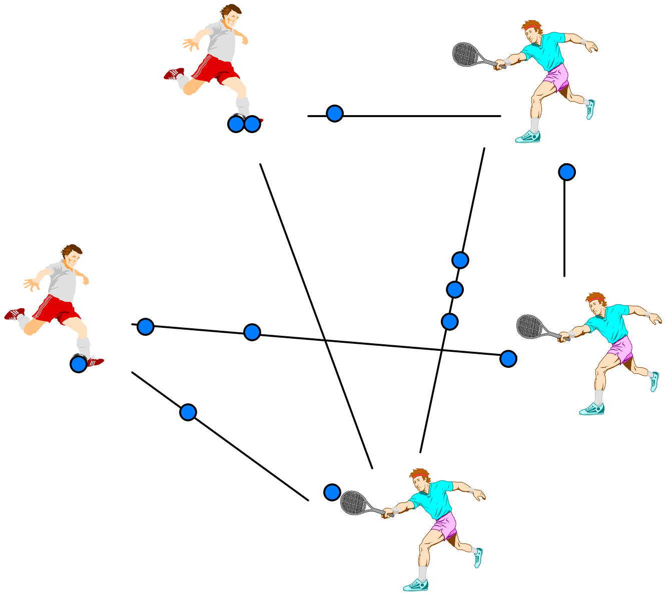
H^+ Cl^- H^+Cl^- HCL

Cl^- HCL H^+

H^+ HCL

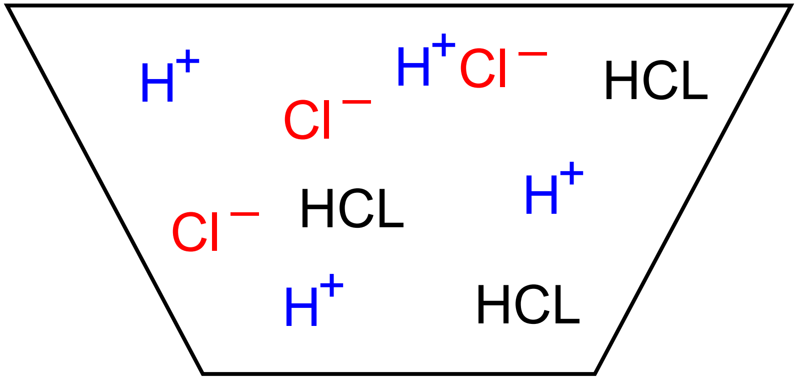
Chemical reaction

$H^+ + Cl^- \longleftrightarrow \mathrm{HCL}$



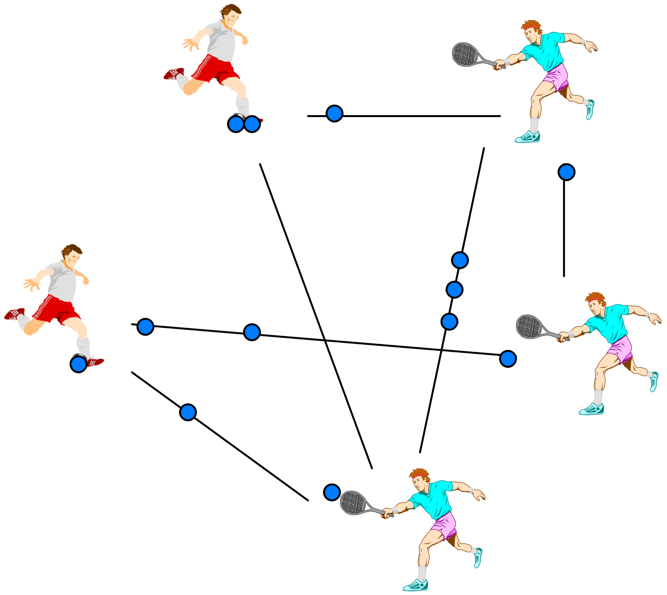
Internet routing

Arbitrary Delay : Brownian Motion



Chemical reaction

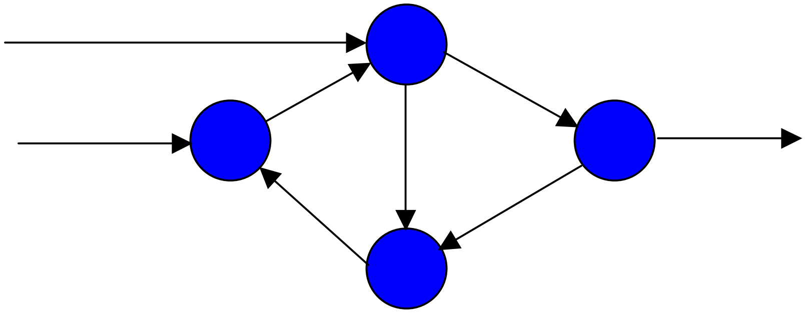
$H^+ + Cl^- \longleftrightarrow HCL$



Internet routing

Models : Kahn networks, π-calculus, CHAM, Join-Calculus, Ambients, etc...

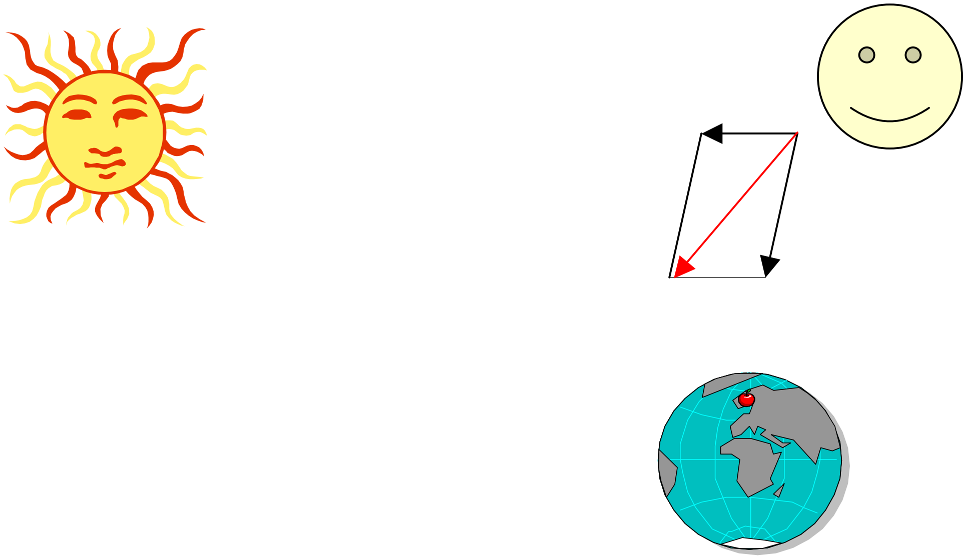
Kahn Networks



nodes = deterministic programs
arrows = infinite fifos

- result-deterministic (independent of computation order)
- easy semantics by flow equations
- heavily used in streaming applications (audio, TV)

Zero delay example: Newtonian Mechanics



Concurrency + Determinism
Calculations are feasible

The most difficult real-time manoeuver ever

Refer to a fabulous drawing of Hergé’s
"On a Marché sur la Lune", in English
"Explorers on the Moon". French edition, page 10,
first drawing.

Drunk Captain Haddock has become a satellite of the Adonis asteroid. To catch him, Tintin, courageously standing on the rocket's side, asked Pr. Calculus to start the rocket's atomic engine. At precisely the right time, he shouts "STOP"!

This is the trickiest real-time manoeuver ever performed by man. It required a perfect understanding of Newtonian Mechanics and absolute synchrony.

The Esterel Runner

```
abort  run Slowly  when  100 Meter ;
```

The Esterel Runner

```
abort  run Slowly  when  100 Meter ;
abort
   every Step do
      run Jump || run Breathe
   end every
when 15 Second ;
```

The Esterel Runner

```
abort  run Slowly  when  100 Meter ;
abort
    every Step do
        run Jump || run Breathe
    end every
when 15 Second ;
run FullSpeed
```

The Esterel Runner

```
loop
   abort  run Slowly  when  100 Meter ;
   abort
      every Step do
         run Jump || run Breathe
      end every
   when 15 Second ;
   run FullSpeed
 each  Lap
```

The Esterel Runner

```
abort
   loop
      abort  run Slowly  when  100 Meter ;
      abort
         every Step do
            run Jump || run Breathe
         end every
      when 15 Second ;
      run FullSpeed
    each  Lap
when  4 Lap
```

The Esterel Runner

```
every Morning do
  abort
    loop
      abort  run Slowly  when  100 Meter ;
      abort
        every Step do
          run Jump || run Breathe
        end every
      when 15 Second ;
      run FullSpeed
    each  Lap
  when  4 Lap
end every
```

The Esterel Runner

```
trap HeartAttack in
    every Morning do
        abort
            loop
                abort  run Slowly  when  100 Meter ;
                abort
                    every Step do
                        run Jump || run Breathe || run CheckHeart
                    end every
                when 15 Second ;                                    exit HeartAttack
                run FullSpeed
            each  Lap
        when  4 Lap
    end every
handle HeartAttack fo
   run RushToHospital
end trap
```

t predictable : vibration

Nothing can illustrate vibration better than Bianca Castafiore, Hergé's famous prima donna. See [1] for details. The power of her voice forcibly shakes the microphone and the ears of the poor spectators.

[1] King's Ottokar Sceptre, Hergé, page 29, last drawing.

propagation of light, electrons, program counter...

Full Abstraction

Bianca Castafiore singing for the King Muskar XII in Klow, Syldavia. King's Ottokar Sceptre, page 38, first drawing.

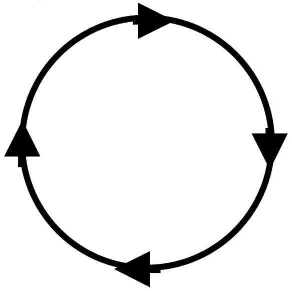
Although the speed of sounds is finite, it is fast enough to look infinite. Full abstraction!

If room is small enough,
predictable delay implements zero-delay

Specify with zero-delay
Implement with predictable delay
Control room size

Software Synchronous Systems

Cycle based



read inputs

compute reaction

produce outputs

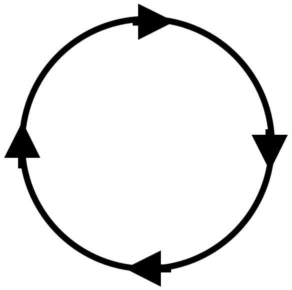
Synchronous = 0-delay = within the same cycle

propagate control

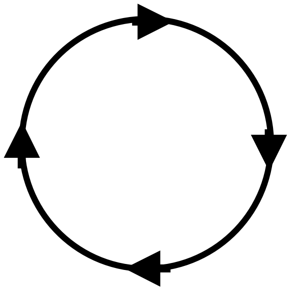
propagate signals

No interference between I/O and computation
Room size control = Worst Case Execution Time (AbsInt)

Concurrency = Cycle Fusion

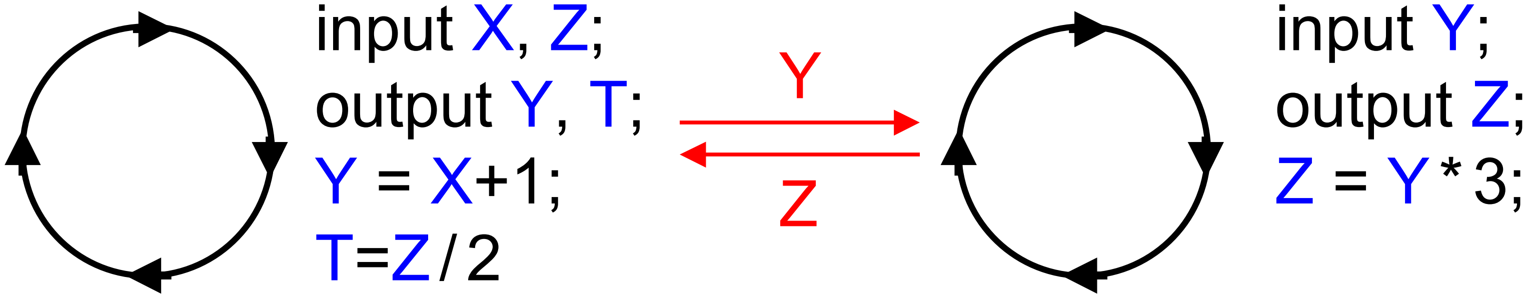


```
input X, Z;
output Y, T;
Y = X+1;
T=Z/2
```



```
input Y;
output Z;
Z = Y*3;
```

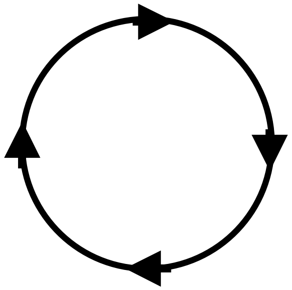
Concurrency = Cycle Fusion



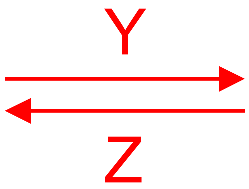
```
input X, Z;
output Y, T;
Y = X+1;
T=Z/2
```

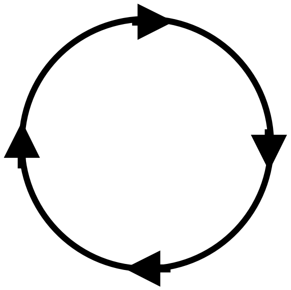
```
input Y;
output Z;
Z = Y*3;
```

Concurrency = Cycle Fusion

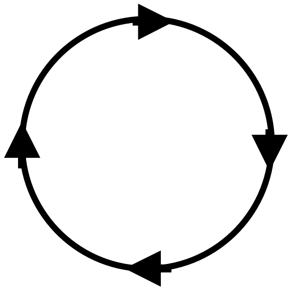


```
input X, Z;
output Y, T;
Y = X+1;
T=Z/2
```



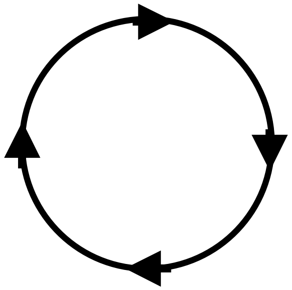


```
input Y;
output Z;
Z = Y*3;
```

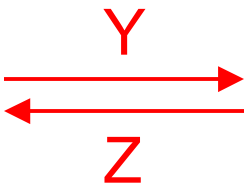


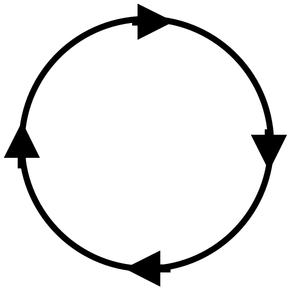
```
input X;
output T;
local Y, Z;
Y = X+1;
Z = Y*3;
T=Z/2;
```

Concurrency = Cycle Fusion

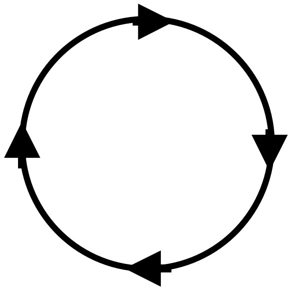


```
input X, Z;
output Y, T;
Y = X+1;
T=Z/2
```





```
input Y;
output Z;
Z = Y*3;
```



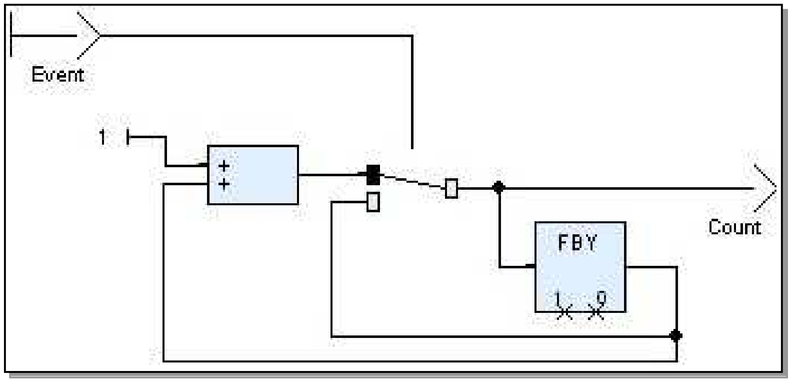
```
input X;
output T;
local Y, Z;
Y = X+1;
Z = Y*3;
T=Z/2;
```

Safe deterministic global variable sharing
No context-switching cost

Lustre = Synchronous Kahn Networks

A simple counter

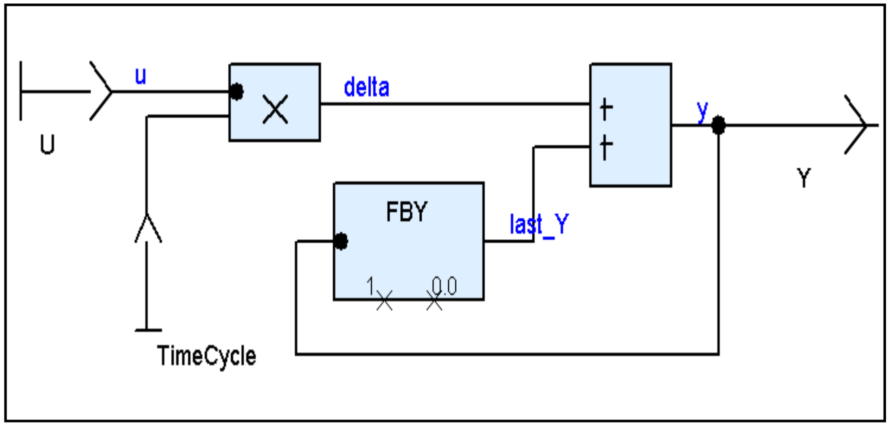
$$\begin{cases} Count(0) = 0 \\ \forall t > 0,\ Count(t) = \begin{cases} Count(t-1)+1, & if\ \ Event(t) = true \\ Count(t-1), & otherwise \end{cases} \end{cases}$$



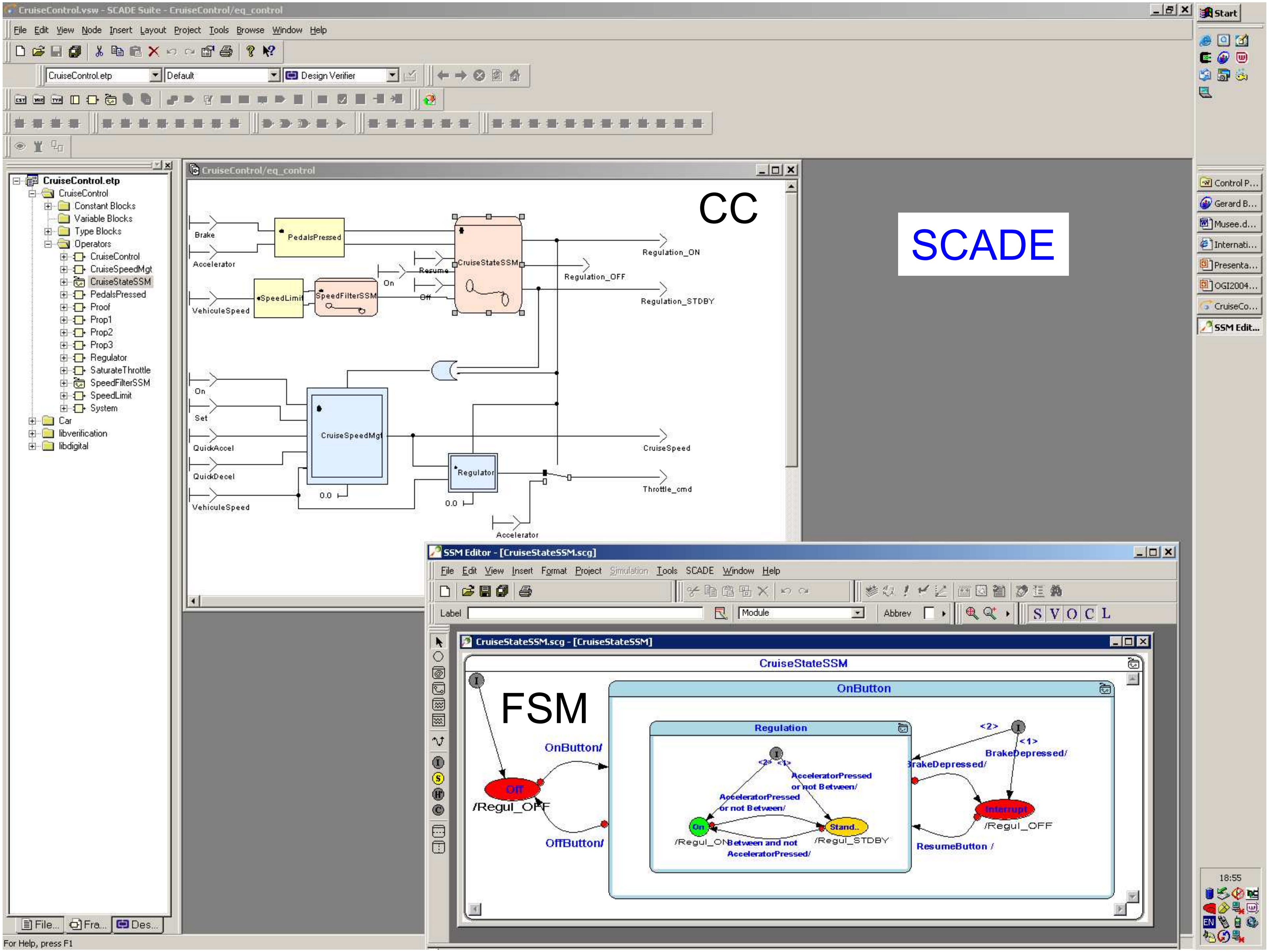
```
Count = 0 ->
   (if Event
    then pre(Count)+1
    else pre(Count))
```

Lustre / Scade Nodes

- A node is a functional module, defined by
  - a formal interface
  - a set of local variables declarations
  - a set of equations
- textual or graphical



```
node Integrator(U: real;
           TimeCycle: real)
returns (Y: real);
    var
        delta : real ;
        last_Y : real;
    delta = u * TimeCycle ;
    y = delta + last_Y ;
    last_Y = fby(y , 1.0 , 0.0)
    ;
```

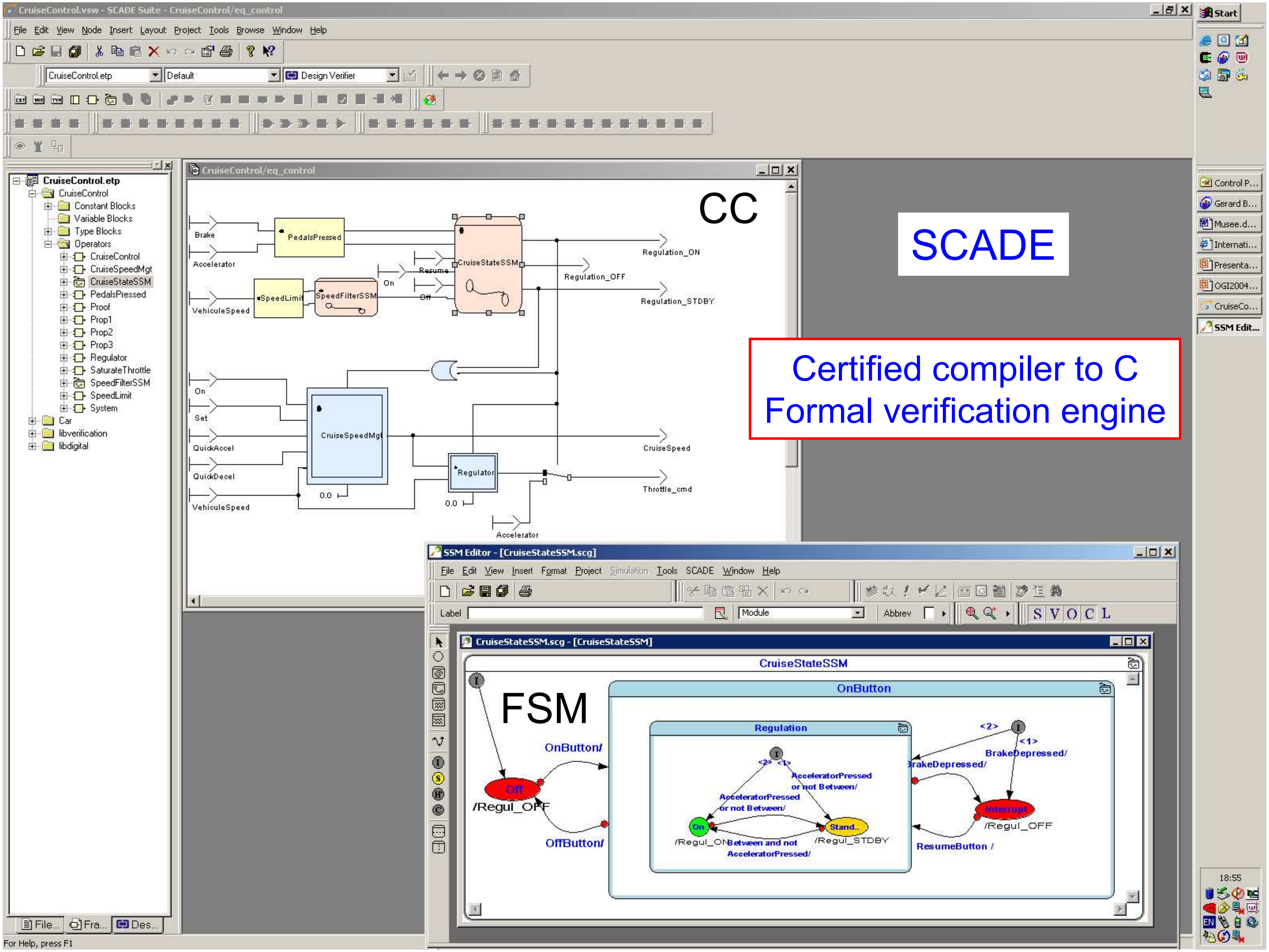


SCADE

CC

FSM

CruiseControl.vsw - SCADE Suite - CruiseControl/eq_control
File Edit View Node Insert Layout Project Tools Browse Window Help
CruiseControl.etp
Default
Design Verifier
CruiseControl.etp
CruiseControl
Constant Blocks
Variable Blocks
Type Blocks
Operators
CruiseControl
CruiseSpeedMgt
CruiseStateSSM
PedalsPressed
Proof
Prop1
Prop2
Prop3
Regulator
SaturateThrottle
SpeedFilterSSM
SpeedLimit
System
Car
libverification
libdigital
File...
Fra...
Des...
For Help, press F1
CruiseControl/eq_control
Brake
Accelerator
PedalsPressed
CruiseStateSSM
Resume
On
Off
VehiculeSpeed
SpeedLimit
SpeedFilterSSM
Regulation_ON
Regulation_OFF
Regulation_STBDY
On
Set
QuickAccel
QuickDecel
VehiculeSpeed
CruiseSpeedMgt
Regulator
CruiseSpeed
Throttle_cmd
0.0
0.0
Accelerator
SSM Editor - [CruiseStateSSM.scg]
File Edit View Insert Format Project Simulation Tools SCADE Window Help
Label
Module
Abbrev
S V O C L
CruiseStateSSM.scg - [CruiseStateSSM]
CruiseStateSSM
OnButton
Regulation
OnButton/
Off
/Regul_OFF
OffButton/
<2> <1>
AcceleratorPressed or not Between/
AcceleratorPressed or not Between/
On
/Regul_ON
Between and not AcceleratorPressed/
Stand..
/Regul_STDBY
<2>
<1>
BrakeDepressed/
BrakeDepressed/
Interrupt
/Regul_OFF
ResumeButton /
Start
Control P...
Gerard B...
Musee.d...
Internati...
Presenta...
OGI2004...
CruiseCo...
SSM Edit...
18:55



CC

SCADE

Certified compiler to C
Formal verification engine

FSM

SCADE Suite™ Customers Base

Civilian Avionics

- Aircraft Braking Systems
- Airbus
- Chengdu Aircraft Development & research Institute
- Chinese Aeronautical Radio Electronics Research Institute
- CMC Electronics Inc.
- Dassault Aviation
- Diehl Avionik Systeme GmH
- Elbit Systems
- Eurocopter
- Honeywell
- Flight Automatic Control Research Institute
- Liebherr-Aerospace
- Messier-Bugatti
- Nanjing University of Aeronautics and Aerospace
- Pratt & Whitney
- Rockwell Collins
- SAAB Aerospace
- SAFRAN
- Seditec
- Smiths Aerospace
- Snecma Aerospace
- Thales Avionics
- Transiciel
- Turbomeca

Specification to safe embedded code

SCADE SUITE™

correct by construction

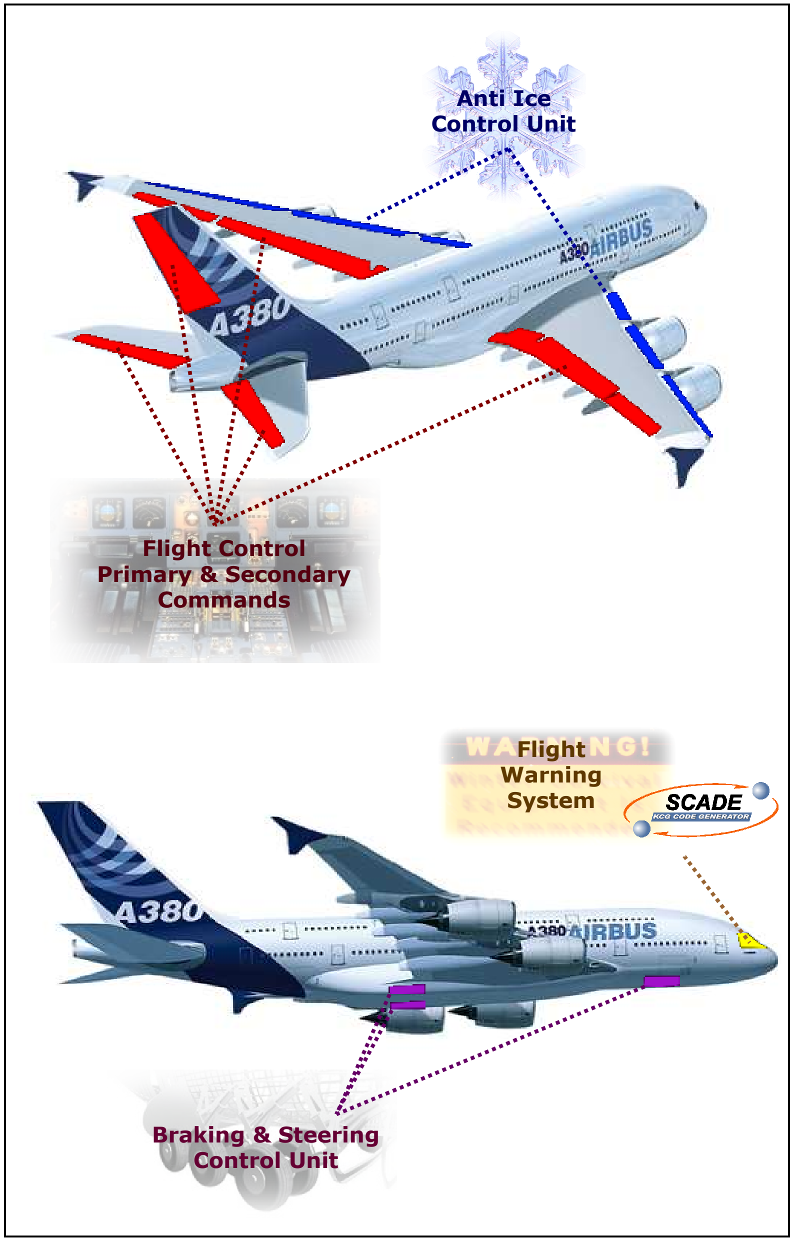
Energy & Transportation

- Ansaldo Signal
- DS & S
- Framatome
- Schneider Electric
- Siemens Transp.

Defense & Space

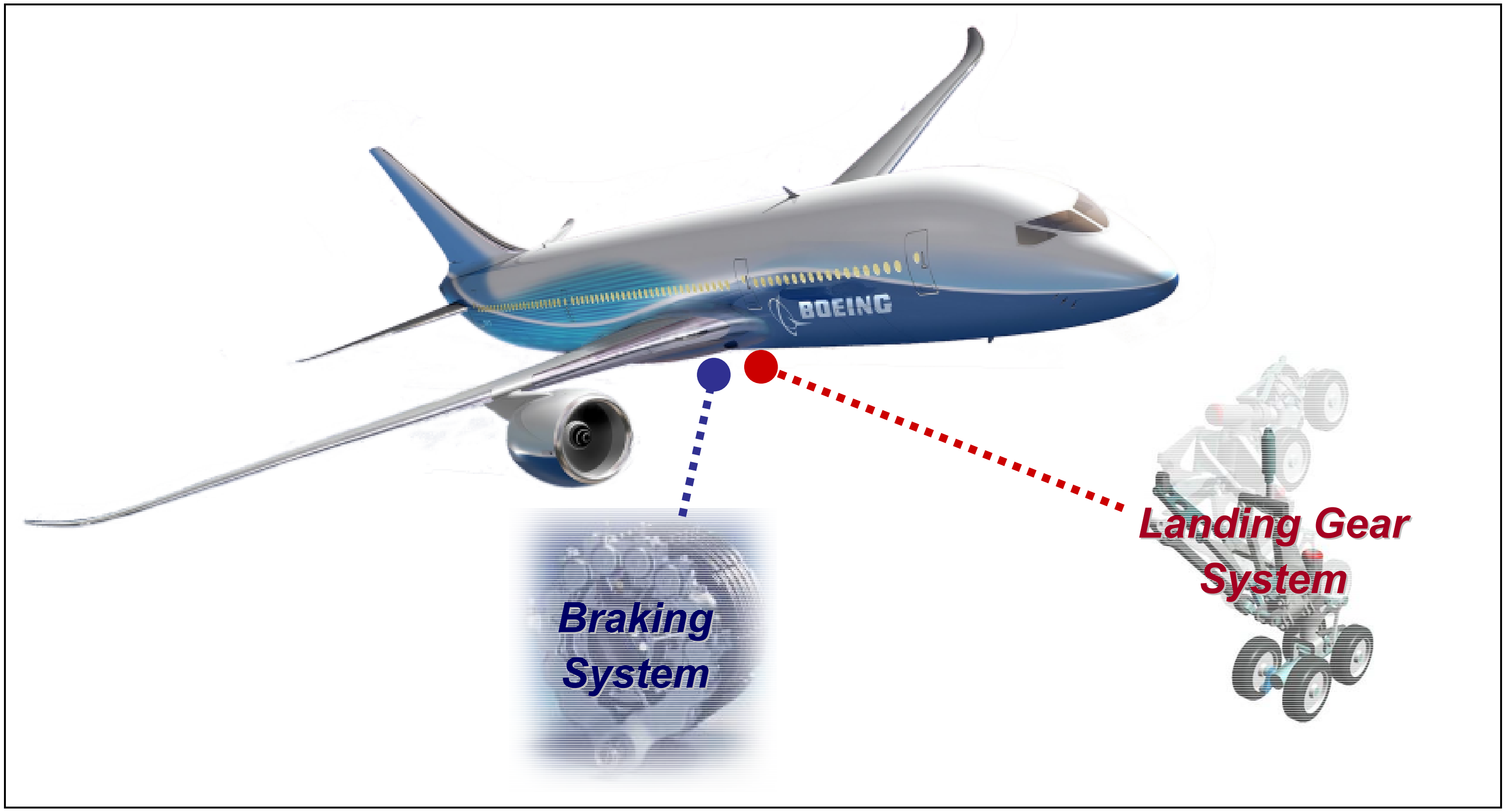
- CAST 504th Institute
- CRIL Technology
- Dassault Aviation
- EADS Military
- EADS SD Electronics
- EADS Space Transportation
- Elbit Systems Ltd.
- ESA
- Eurocopter
- Hills US Air Force Base
- Hispano-Suiza
- Intertechnique
- Lockheed Martin
- MBDA
- NASA
- Rockwell Collins
- Rockwell Collins Flight Dynamics
- SAGEM
- Thales Airborne Systems
- Thales Communication

SCADE Suite in the A380



- SCADE = Airbus corporate standard for all new airplanes developments
  - Flight Control system
  - Flight Warning system
  - Electrical Load Management system
  - Anti Icing system
  - Braking and Steering system
  - Cockpit Display system
  - Part of ATSU (Board / Ground comms)
  - FADEC (Engine Control)
  - EIS2 : Specification GUI Cockpit:
    - PFD : Primary Flight Display
    - ND : Navigation Display
    - EWD : Engine Warning Display
    - SD : System Display

SCADE Suite in the 787



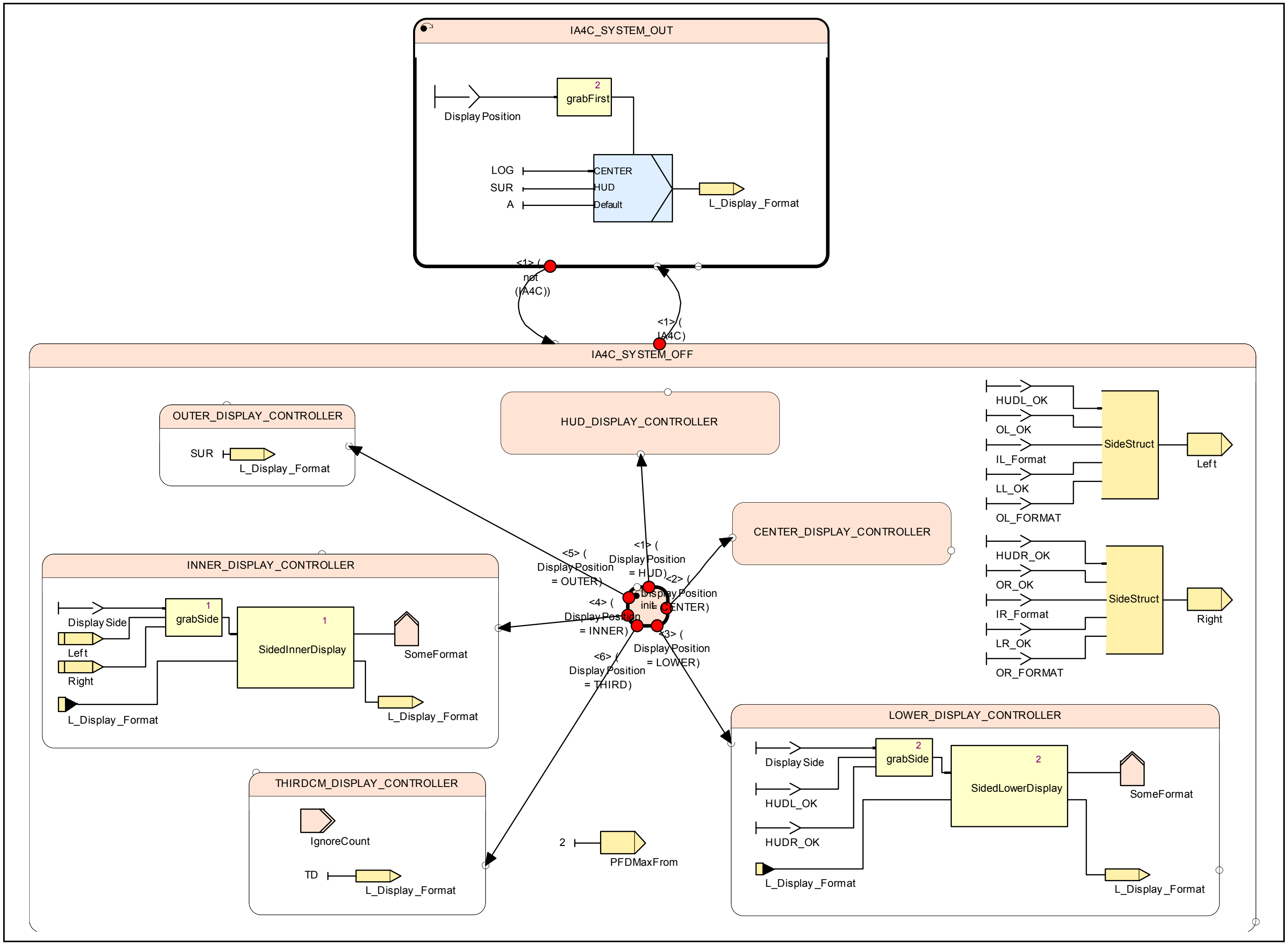
- Landing Gear System *(Smiths Aerospace)*
- Braking System *(Messier Bugatti)*

EUROCOPTER

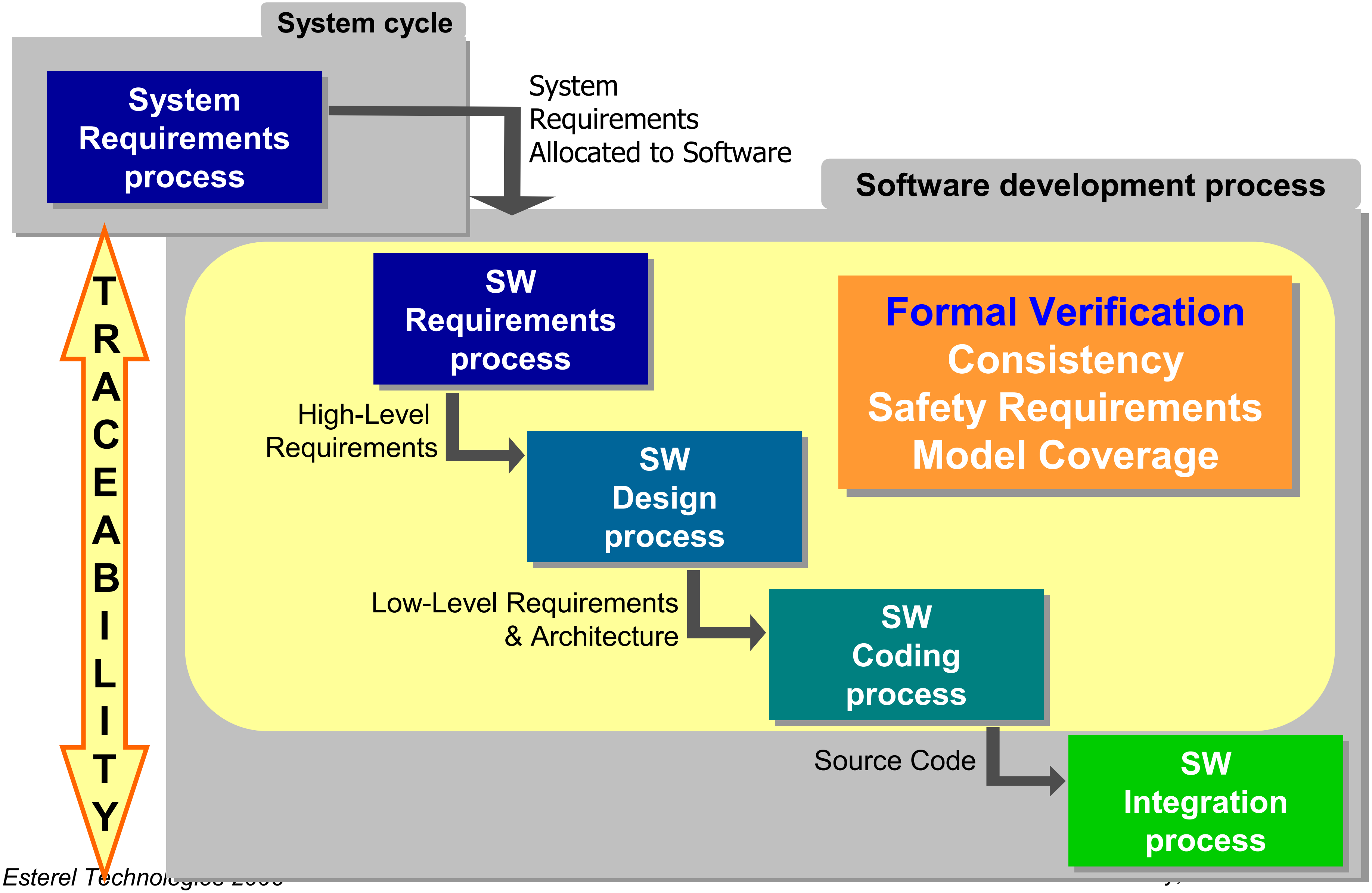
- World leader in civilian helicopters
- Introduced SCADE Suite™ for **EC135 and EC155 autopilots**
- Results
  - 90% of the code with SCADE
  - Development time divided by 2
  - (8 level A certifications by JAA : EC155, EC135, EC145; EC225 on-going)
  - The entire modification cycle can be performed in less than 48 h !

Code Generator for
DO-178B
Level-A
Certification

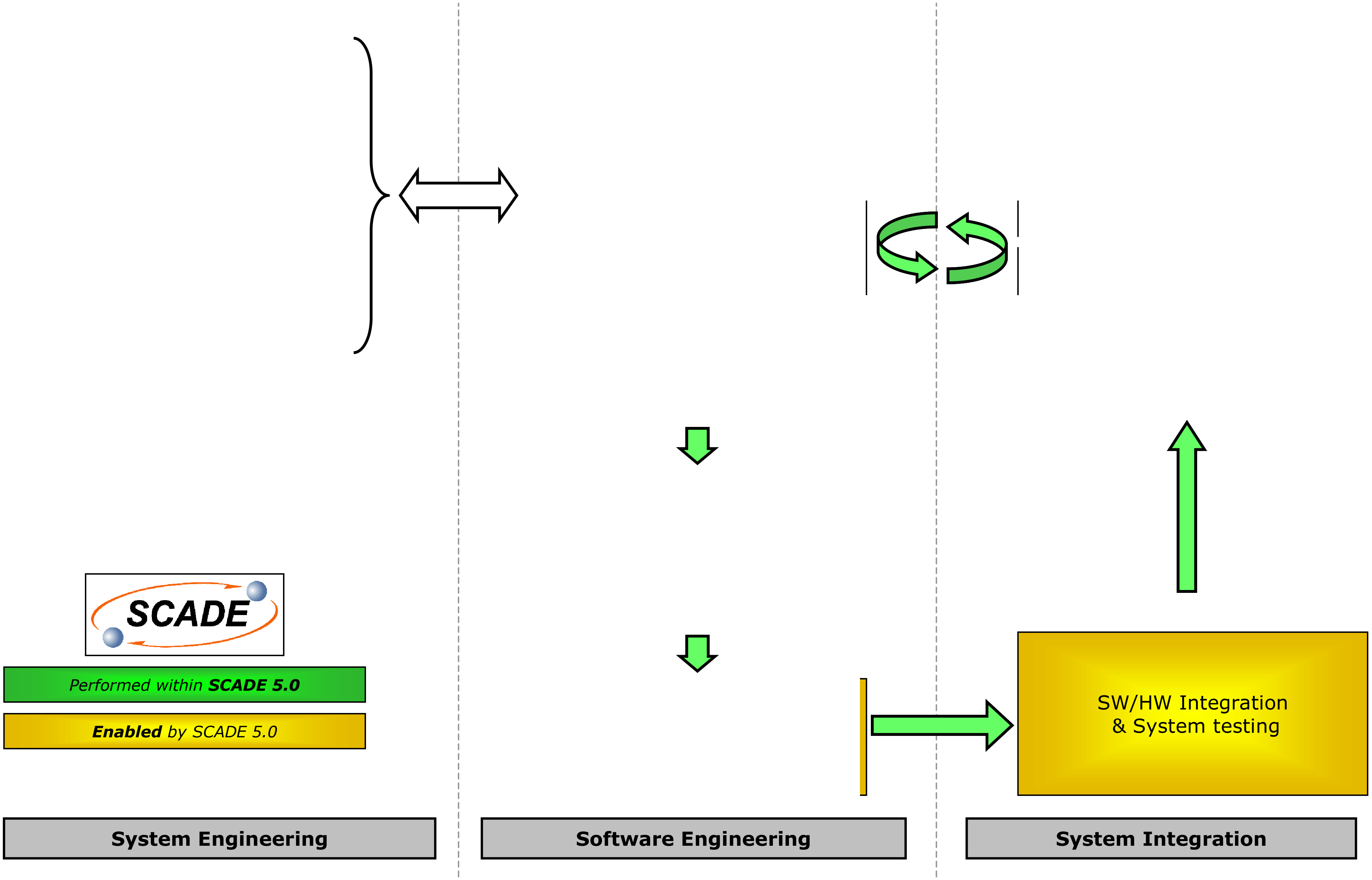
SCADE 6 : full data-flow / control-flow integration



DO-178B Development with SCADE



SCADE System & Software Design flow



SCADE

Performed within ***SCADE 5.0***

Enabled *by SCADE 5.0*

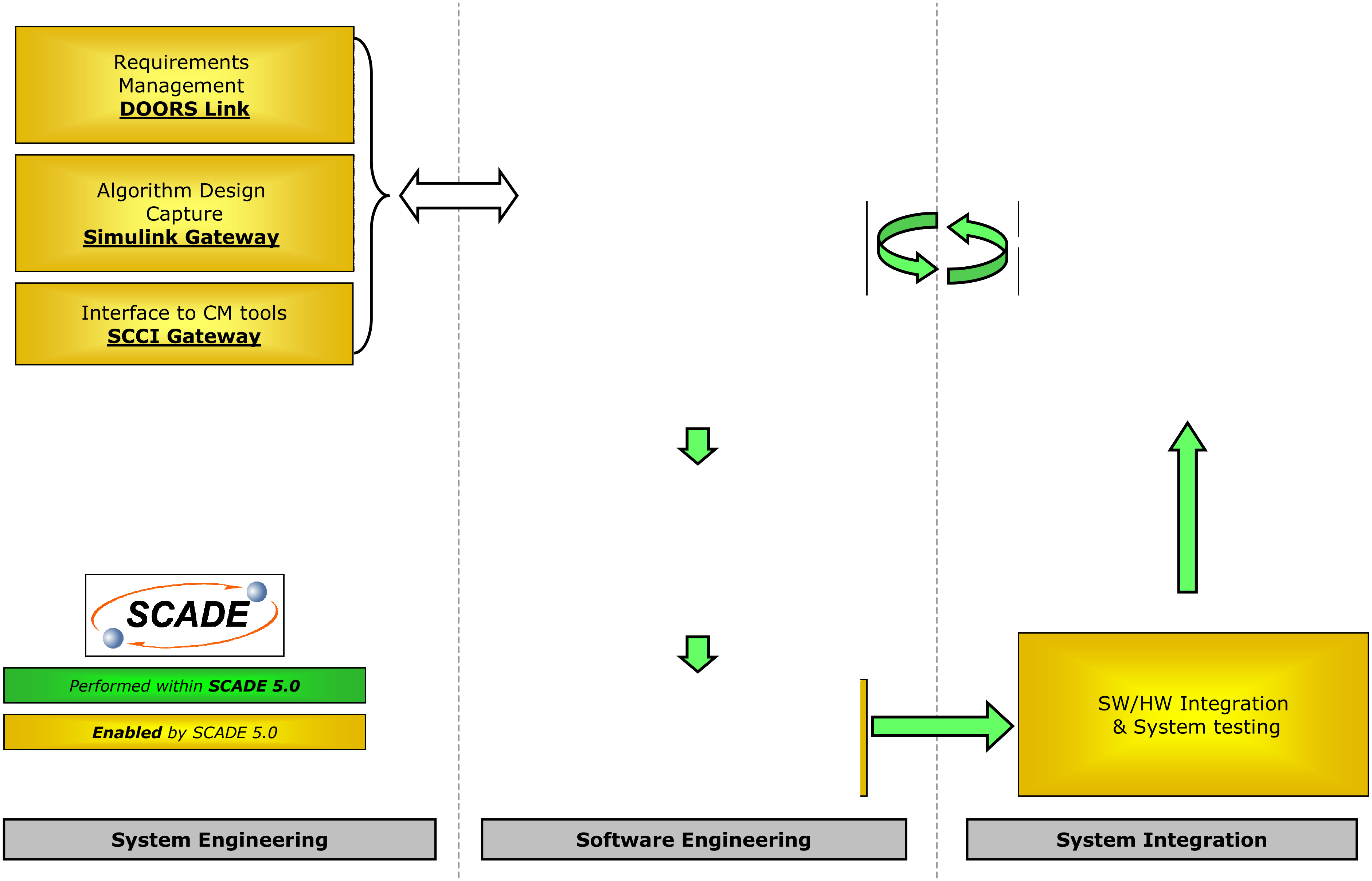
SW/HW Integration & System testing

System Engineering

Software Engineering

System Integration

SCADE System & Software Design flow



Requirements Management **DOORS Link**

Algorithm Design Capture **Simulink Gateway**

Interface to CM tools **SCCI Gateway**

SCADE

Performed within ***SCADE 5.0***

Enabled *by SCADE 5.0*

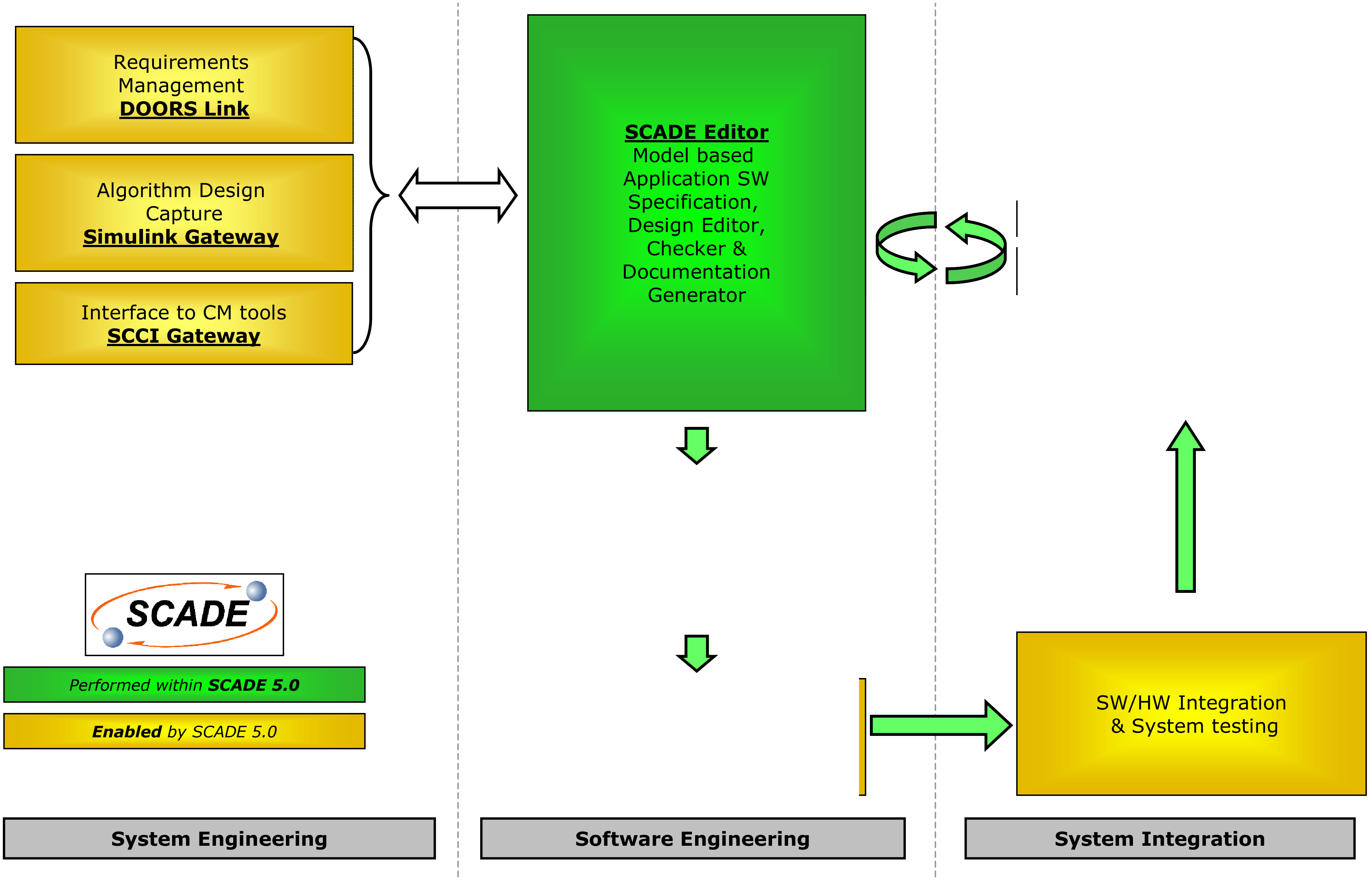
SW/HW Integration & System testing

System Engineering

Software Engineering

System Integration

SCADE System & Software Design flow



Requirements Management
DOORS Link

Algorithm Design Capture
Simulink Gateway

Interface to CM tools
SCCI Gateway

SCADE Editor
Model based Application SW Specification, Design Editor, Checker & Documentation Generator

SW/HW Integration & System testing

Performed within ***SCADE 5.0***

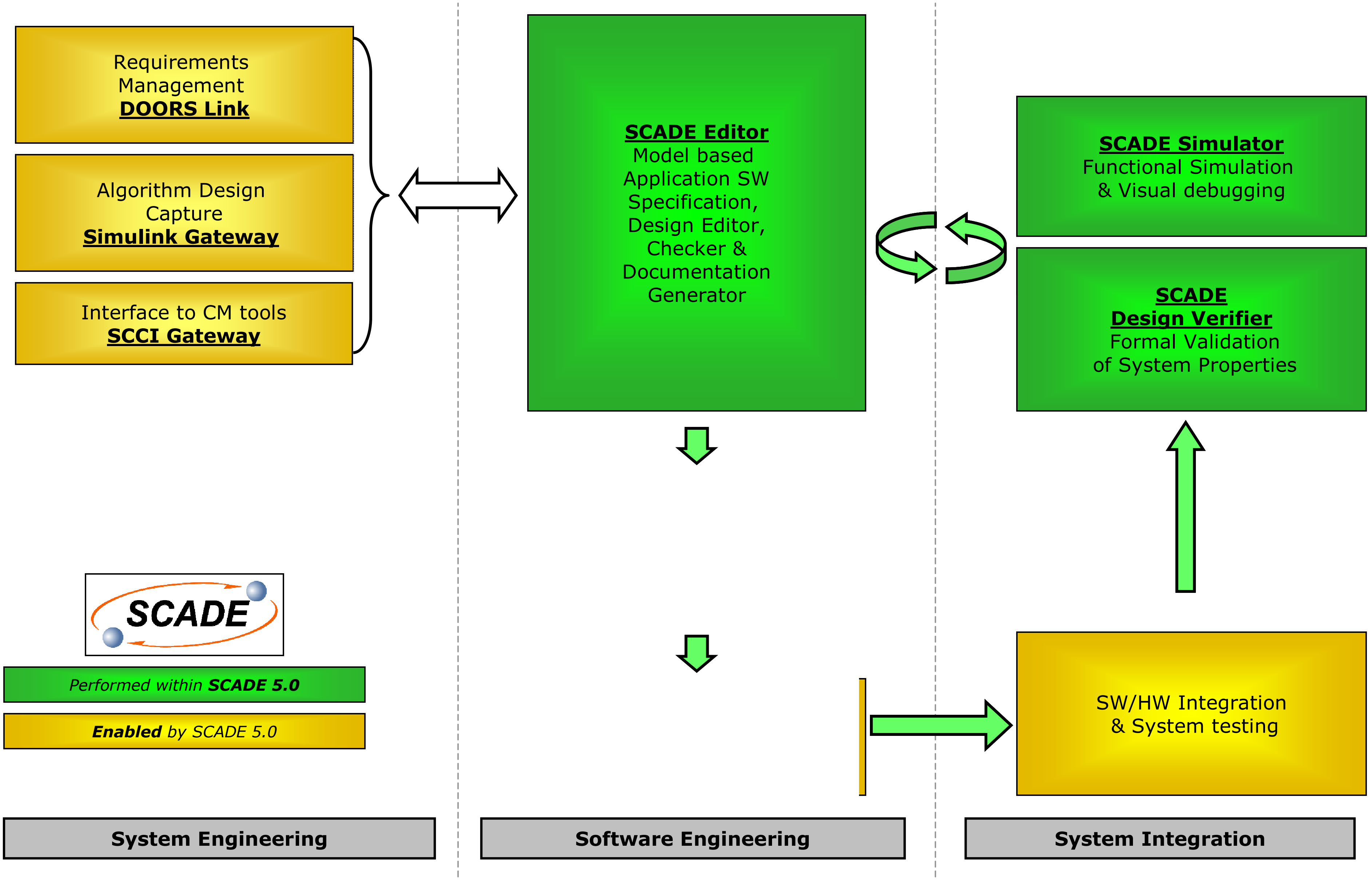
Enabled *by SCADE 5.0*

System Engineering

Software Engineering

System Integration

SCADE System & Software Design flow



Requirements Management
DOORS Link

Algorithm Design Capture
Simulink Gateway

Interface to CM tools
SCCI Gateway

SCADE Editor
Model based Application SW Specification, Design Editor, Checker & Documentation Generator

SCADE Simulator
Functional Simulation & Visual debugging

SCADE Design Verifier
Formal Validation of System Properties

SW/HW Integration & System testing

SCADE

Performed within ***SCADE 5.0***

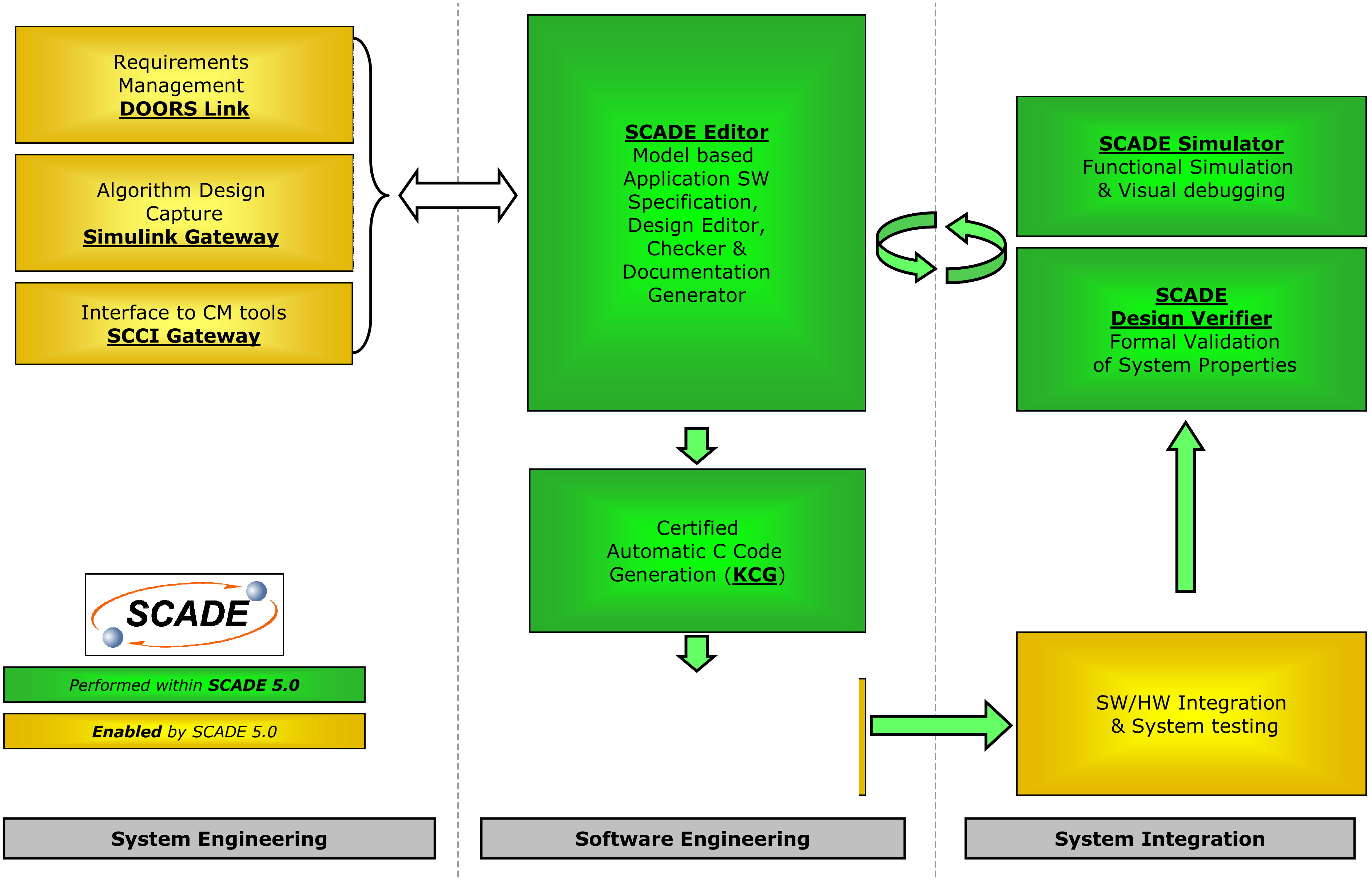
Enabled *by SCADE 5.0*

System Engineering

Software Engineering

System Integration

SCADE System & Software Design flow



Requirements Management
DOORS Link

Algorithm Design Capture
Simulink Gateway

Interface to CM tools
SCCI Gateway

SCADE Editor
Model based Application SW Specification, Design Editor, Checker & Documentation Generator

Certified Automatic C Code Generation (**KCG**)

SCADE Simulator
Functional Simulation & Visual debugging

SCADE Design Verifier
Formal Validation of System Properties

SW/HW Integration & System testing

SCADE

*Performed within **SCADE 5.0***

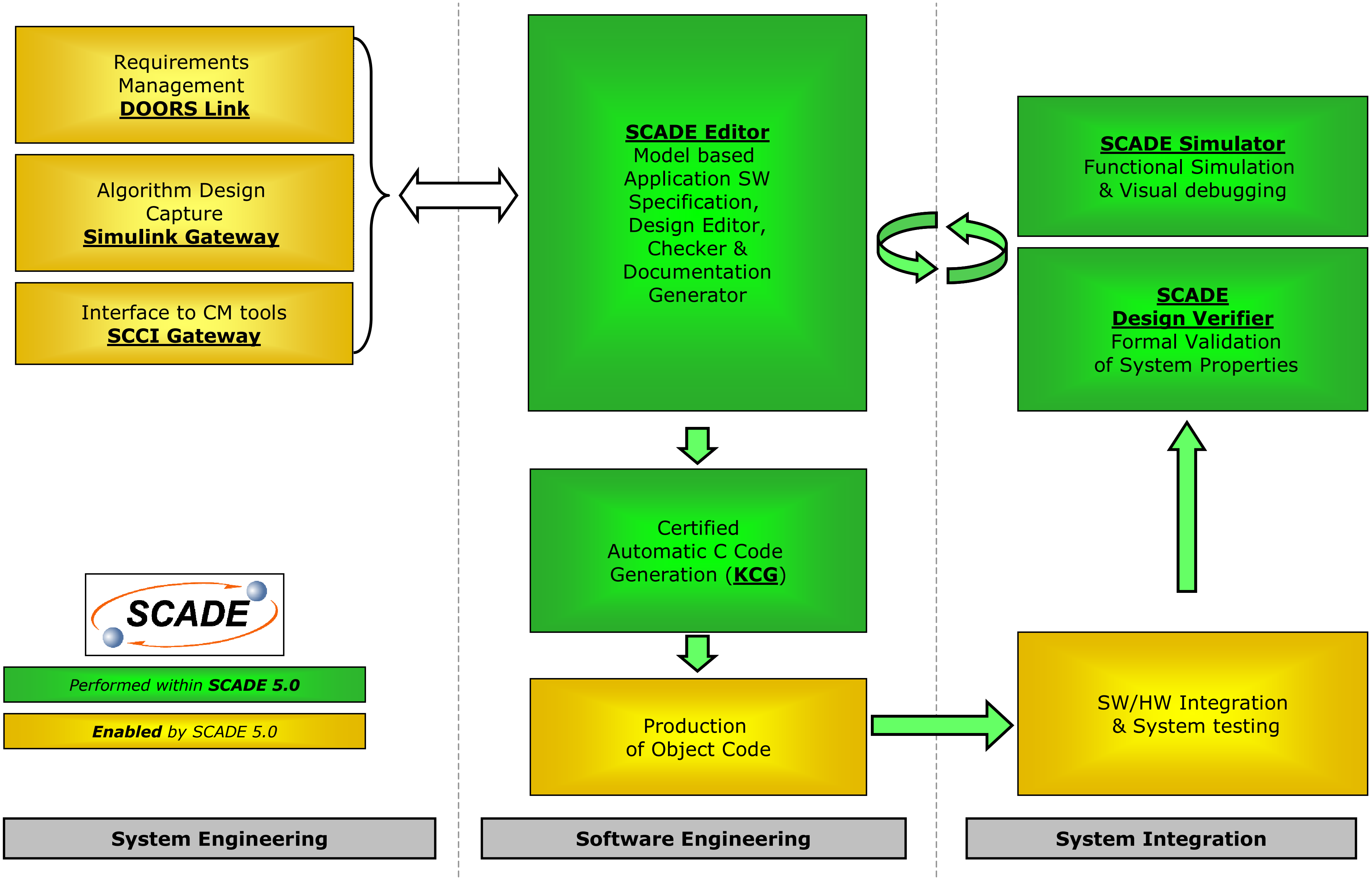
***Enabled** by SCADE 5.0*

System Engineering

Software Engineering

System Integration

SCADE System & Software Design flow



Requirements Management
DOORS Link

Algorithm Design Capture
Simulink Gateway

Interface to CM tools
SCCI Gateway

SCADE Editor
Model based Application SW Specification, Design Editor, Checker & Documentation Generator

SCADE Simulator
Functional Simulation & Visual debugging

SCADE Design Verifier
Formal Validation of System Properties

Certified Automatic C Code Generation (**KCG**)

Production of Object Code

SW/HW Integration & System testing

Performed within ***SCADE 5.0***

Enabled *by SCADE 5.0*

System Engineering

Software Engineering

System Integration

Automotive specifics

- Application domains

  engine control

  suspension, braking, airbags, locking system, A/C, etc.

  entertainment systems

- Technical features

  Misra compliant C code generation

  Links with OSEK layers

  Links with TTA / FlexRay time-triggered networks

  Fixed-point implementer

  IEC 61508 certification for all SIL levels

Testing Support

- Model test coverage (MTC)

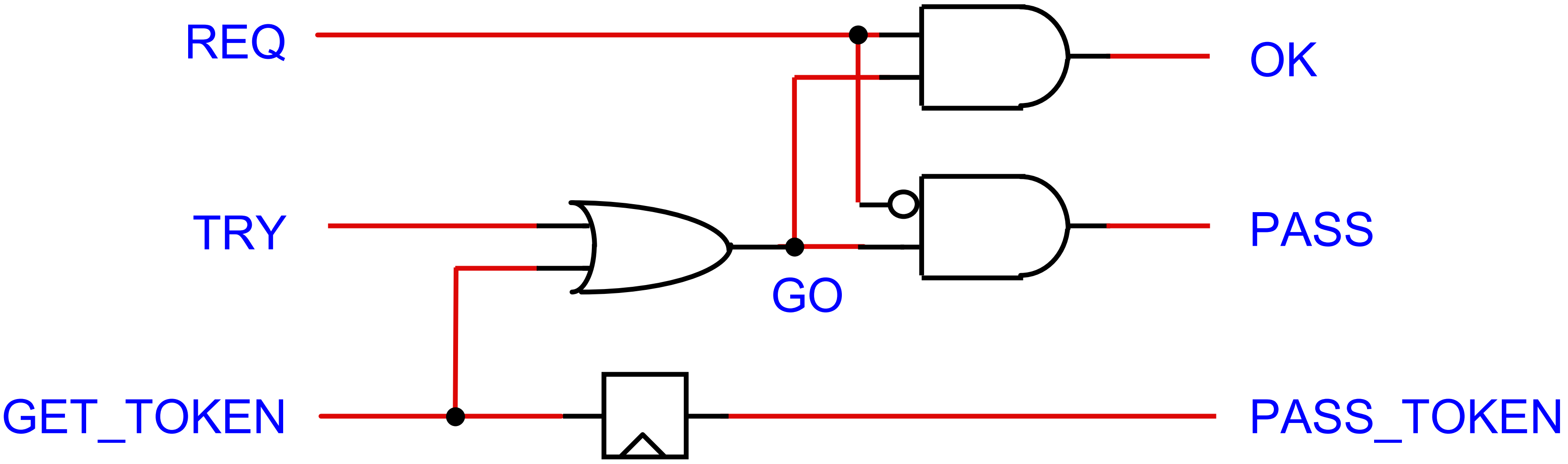
  establishes test suite coverage w.r.t. model

  checks coverage of operator boundary cases

- Compiler Verification Kit (CVK)

  exhaustive check of target C compiler for all SCADE-generated C patterns

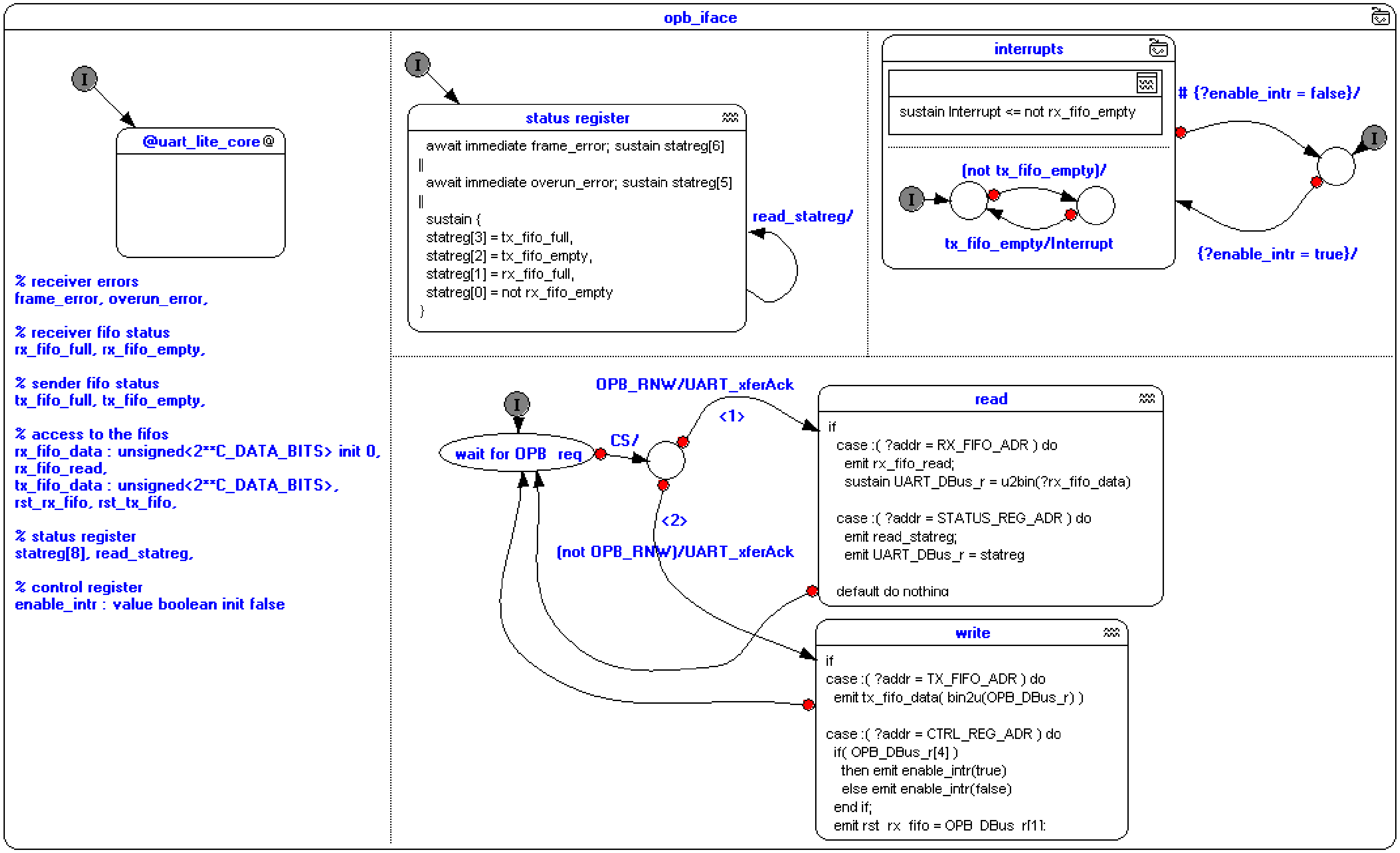
Hardware Synchrony: the RTL model



OK = REQ and GO
PASS = not REQ and GO
GO = TRY or GET_TOKEN
PASS_TOKEN = reg(GET_TOKEN)

Room size control = timing closure

Esterel v7 (Berry – Kishinevsky)



```
% receiver errors
frame_error, overun_error,

% receiver fifo status
rx_fifo_full, rx_fifo_empty,

% sender fifo status
tx_fifo_full, tx_fifo_empty,

% access to the fifos
rx_fifo_data : unsigned<2**C_DATA_BITS> init 0,
rx_fifo_read,
tx_fifo_data : unsigned<2**C_DATA_BITS>,
rst_rx_fifo, rst_tx_fifo,

% status register
statreg[8], read_statreg,

% control register
enable_intr : value boolean init false
```

status register

```
await immediate frame_error; sustain statreg[6]
||
await immediate overun_error; sustain statreg[5]
||
sustain {
statreg[3] = tx_fifo_full,
statreg[2] = tx_fifo_empty,
statreg[1] = rx_fifo_full,
statreg[0] = not rx_fifo_empty
}
```

```
sustain Interrupt <= not rx_fifo_empty
```

read

```
if
 case :( ?addr = RX_FIFO_ADR ) do
  emit rx_fifo_read;
  sustain UART_DBus_r = u2bin(?rx_fifo_data)

 case :( ?addr = STATUS_REG_ADR ) do
  emit read_statreg;
  emit UART_DBus_r = statreg

 default do nothing
```

write

```
if
case :( ?addr = TX_FIFO_ADR ) do
  emit tx_fifo_data( bin2u(OPB_DBus_r) )

case :( ?addr = CTRL_REG_ADR ) do
  if( OPB_DBus_r[4] )
   then emit enable_intr(true)
   else emit enable_intr(false)
  end if;
  emit rst_rx_fifo = OPB_DBus_r[1];
```

text + graphics, concurrency + sequencing
clear semantics

Esterel Consortium

- In 2001 Esterel Technologies formed a consortium of leading Semiconductor companies
  - Early adopters of Esterel Studio™
  - Strategic involvement
  - Collaborative specification of the requirements
  - Strong influence on roadmap
  - Working with Esterel Technologies for IEEE standardization of the Esterel language
  - And more recently…

ST Microelectronics

Texas Instruments

PHILIPS

axalto
A Schlumberger company

intel®

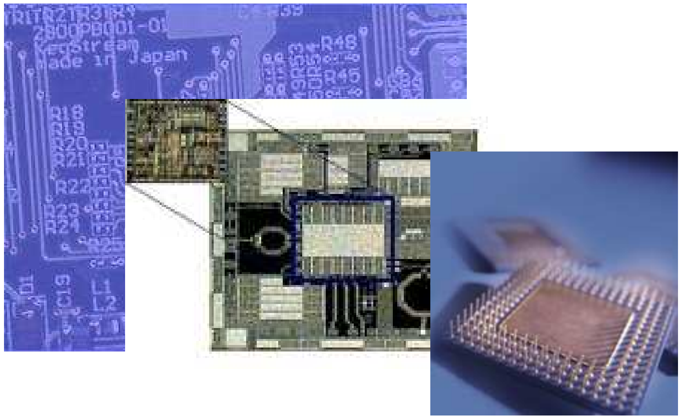
SAMSUNG

eve

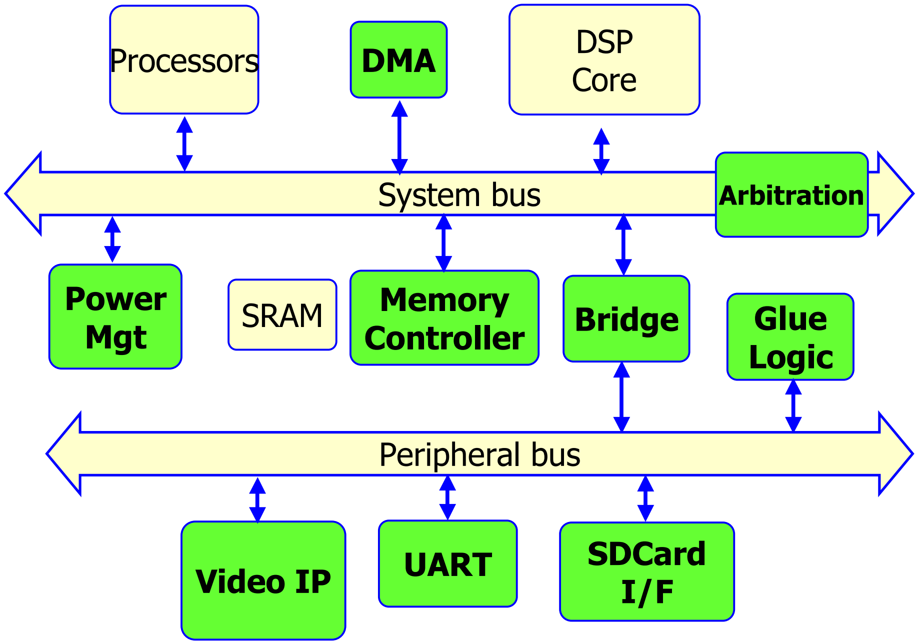
ARM

Application Targets

- Bus interfaces and peripheral controllers
  - Bus Bridge
  - Serial ATA
  - Secure Memory Card
  - Video Controller



- Processor core peripherals
  - Complex Instruction and Data Cache
  - Arbiters
  - Complex Power Management
  - DMA
  - Interrupt Controller

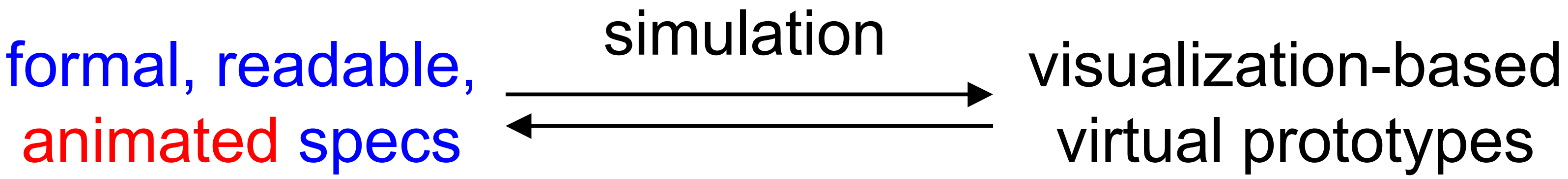


- Communication IPs
  - Serial Controller
  - HDLC
  - Fast serial links (UART, Aurora)
  - Bluetooth Call Control
  - Ethernet MAC Controller

The Usage Model

formal, readable,
animated specs

The Usage Model

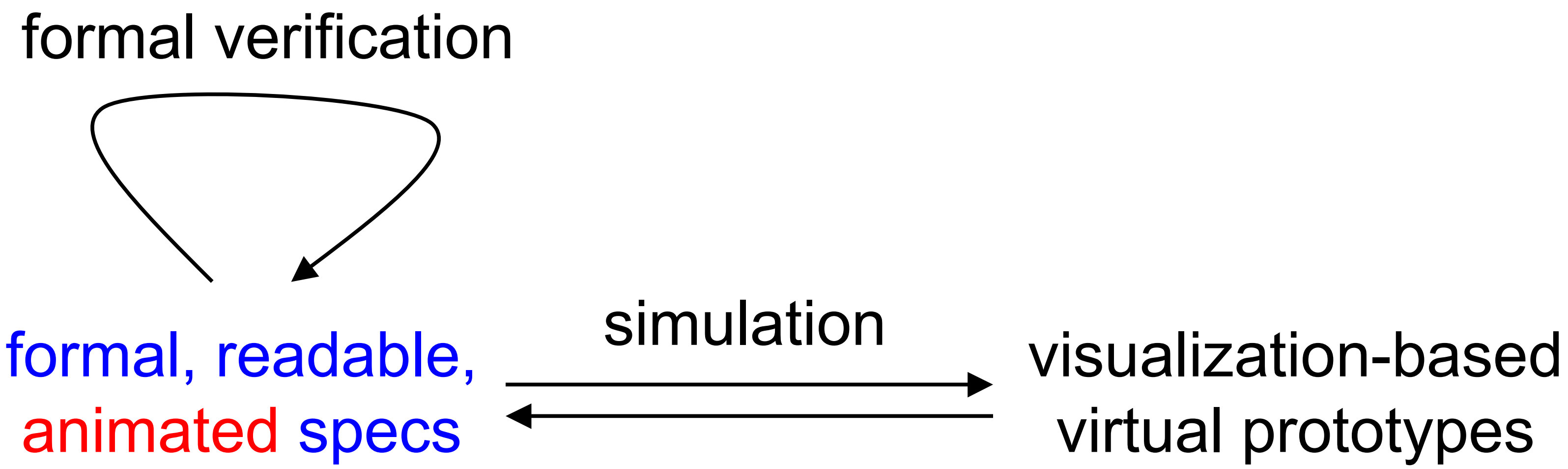


formal, readable, animated specs

simulation

visualization-based virtual prototypes

The Usage Model



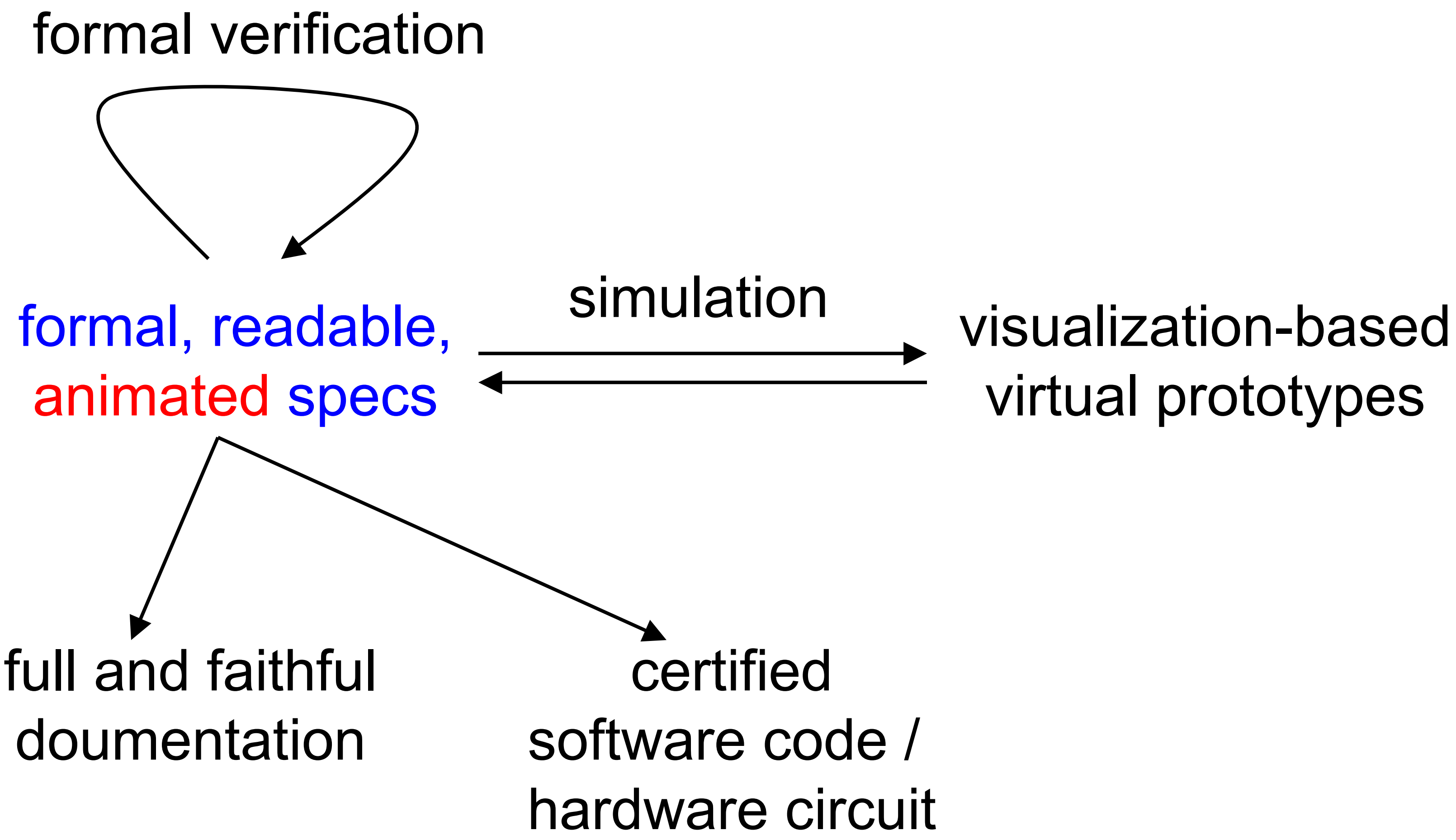
formal verification

formal, readable, animated specs

simulation

visualization-based virtual prototypes

The Usage Model



formal verification

formal, readable, animated specs

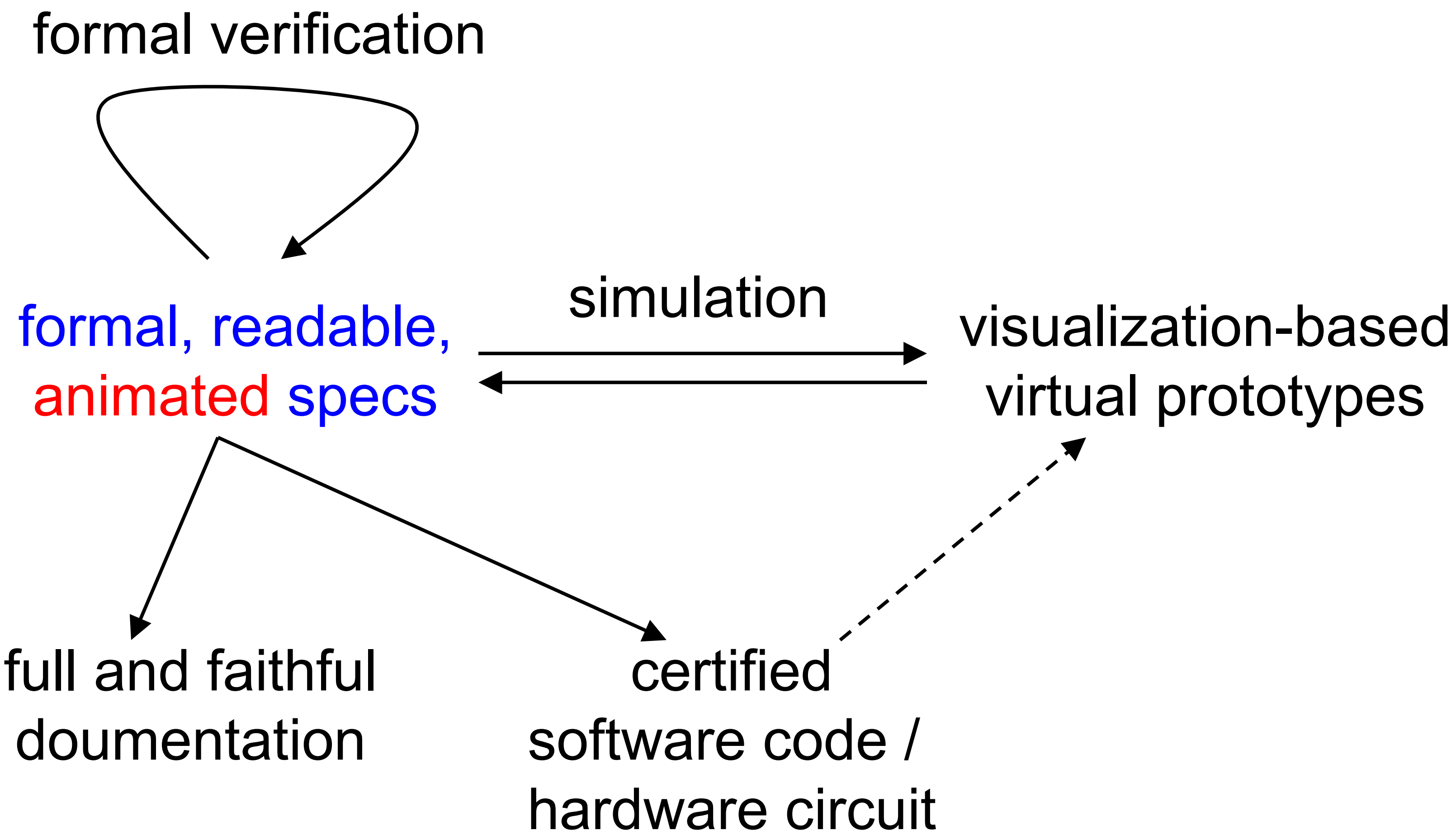
simulation

visualization-based virtual prototypes

full and faithful doumentation

certified software code / hardware circuit

The Usage Model



formal verification

formal, readable, animated specs

simulation

visualization-based virtual prototypes

full and faithful doumentation

certified software code / hardware circuit

Computer Science at Work

Computer Science at Work

1. Language design & mathematical semantics
   - Esterel: imperative, SOS semantics (residuals)
     - constructive logic, proof networks
   - Lustre/SCADE: declarative, functional, denotational semantics
     - clock calculus = static type-check of dynamics

Computer Science at Work

1. Language design & mathematical semantics

    Esterel: imperative, SOS semantics (residuals)

        constructive logic, proof networks

    Lustre/SCADE: declarative, functional, denotational semantics

        clock calculus = static type-check of dynamics

2. Compiling

    Esterel: translation to circuits (hardware)

        translation to concurrent flow graphs (software)

    Lustre/SCADE: clock calculus to drive expression execution

    All: static scheduling of elementary actions

Computer Science at Work

1. Language design & mathematical semantics

    Esterel: imperative, SOS semantics (residuals)

        constructive logic, proof networks

    Lustre/SCADE: declarative, functional, denotational semantics

        clock calculus = static type-check of dynamics

2. Compiling

    Esterel: translation to circuits (hardware)

        translation to concurrent flow graphs (software)

    Lustre/SCADE: clock calculus to drive expression execution

    All: static scheduling of elementary actions

3. Formal Verification: properties and equivalence

    forward / backward reachable state space analysis (BDDs)

    SAT + numerical solving

    Abstract Interpretation (Astrée, Cousot)

Research Directions

Research Directions

- Scale verification techniques
  - improve SAT / numerical / abstract interpretation engines
  - develop assume / guarantee verification
  - prove compilers correct: Schneider (HOL), Leroy (Coq)

Research Directions

- Scale verification techniques
  - improve SAT / numerical / abstract interpretation engines
  - develop assume / guarantee verification
  - prove compilers correct: Schneider (HOL), Leroy (Coq)

- Extend model to distributed systems (castles instead of rooms)

Research Directions

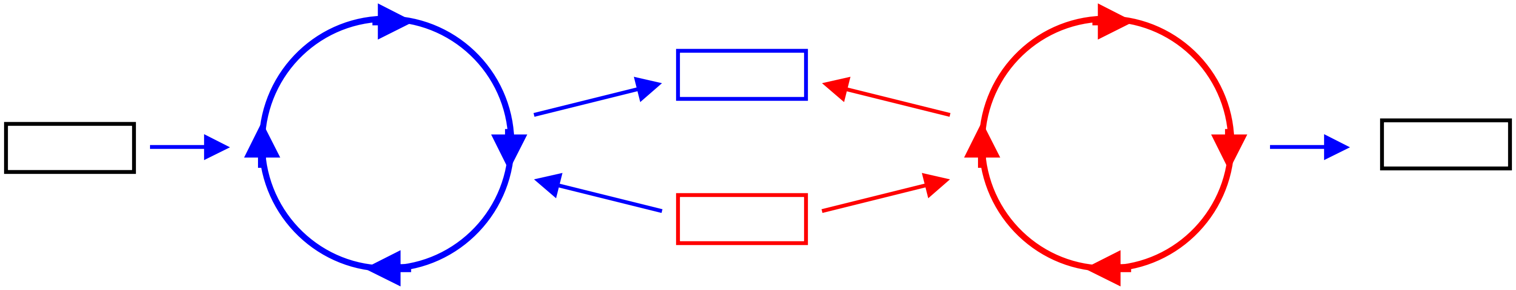
- Scale verification techniques
    - improve SAT / numerical / abstract interpretation engines
    - develop assume / guarantee verification
    - prove compilers correct: Schneider (HOL), Leroy (Coq)

- Extend model to distributed systems (castles instead of rooms)

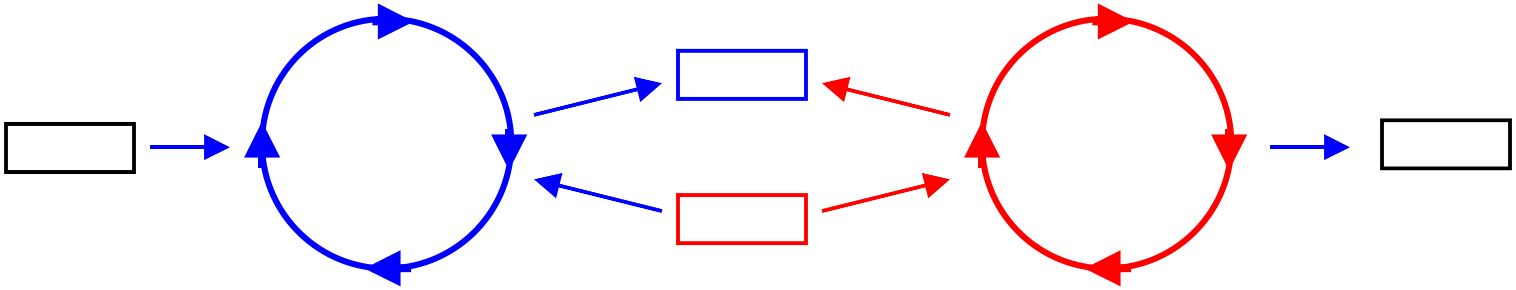
=>Add a controllable amount of asynchrony

- timed-triggered networks (Kopetz, TTP, FlexRay)
- distributed sampling / Nyquist theorem (Caspi)
- elastic circuits (Cortadella & Kishinevsky)
- ...

Distribution by Mutual Sampling



Distribution by Mutual Sampling



- Works because of control theory stability results, not because of computer science ones (Caspi & al.)

- Similar to multiclock hardware, but much simpler (no metastability issues)

Conclusion

- Synchrony is much simpler than asynchrony
  - manageable large-scale concurrency + sequencing
  - equally good for software an hardware
- Synchronous formal methods are used in industry
  - formal languages
  - formal compilation schemes
  - formal verification
- Computing on formal programs is the future
  - automatic / semi-manual property verification
  - compiler correctness proof

Conclusion

- Synchrony is much simpler than asynchrony
  - manageable large-scale concurrency + sequencing
  - equally good for software an hardware
- Synchronous formal methods are used in industry
  - formal languages
  - formal compilation schemes
  - formal verification
- Computing on formal programs is the future
  - automatic / semi-manual property verification
  - compiler correctness proof

Get Esterel Studio and SCADE
they are free
for teaching and academic usage