The Evolution of the Synchronous Model

Gérard Berry

Professor at Collège de France
Chief Scientist

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

Safety-critical certified embedded software

Avionics, railways, heavy industry, automotive

Products: SCADE Suite, Scade Display

Language: Scade 6

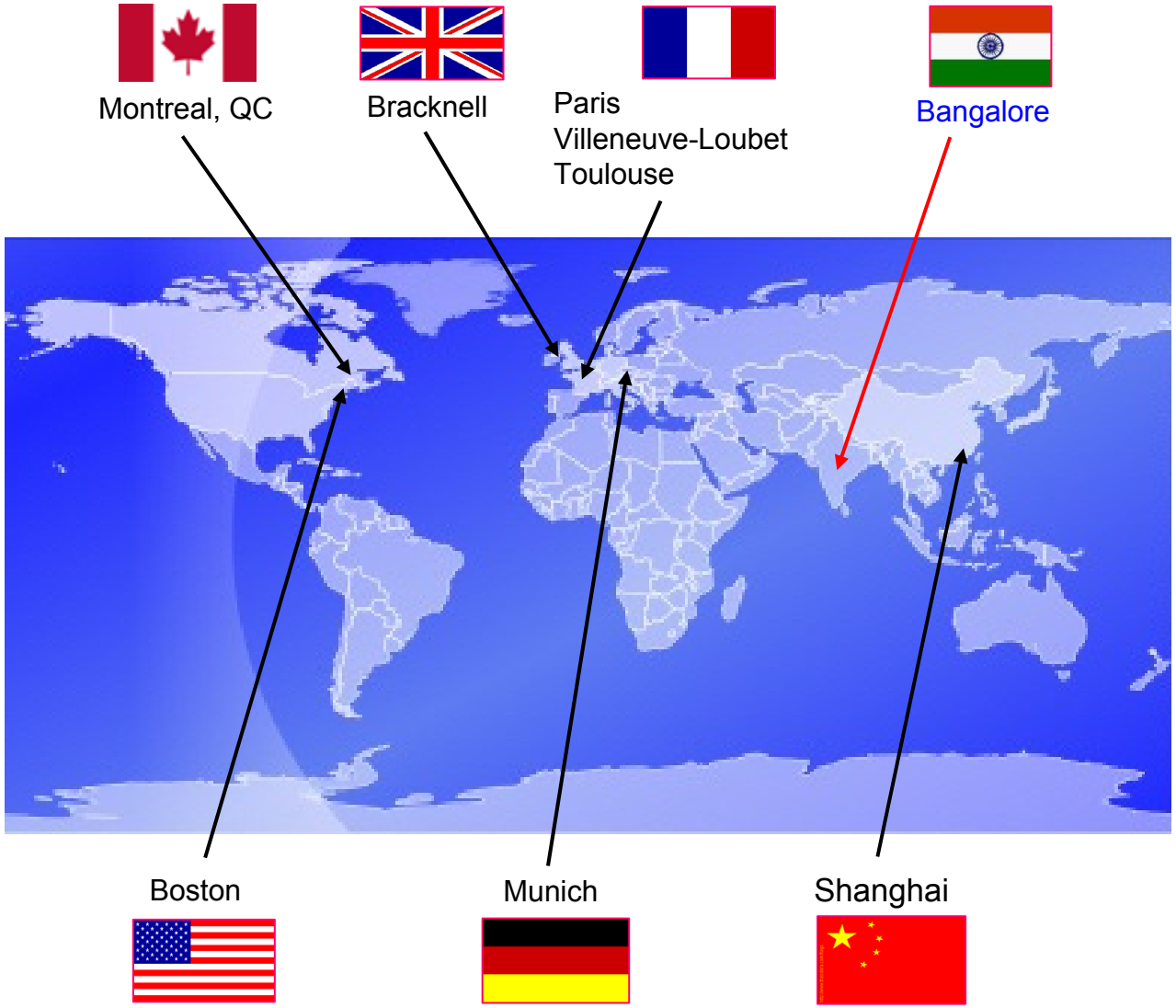
Circuit synthesis and verification

Consumer electronics

Product: Esterel Studio

Language : Esterel v7

140 people, 7 countries, 150 customers



+ Partner Network:

India, Israel,
Japan, Russia…

SCADE Aerospace & Defense Applications

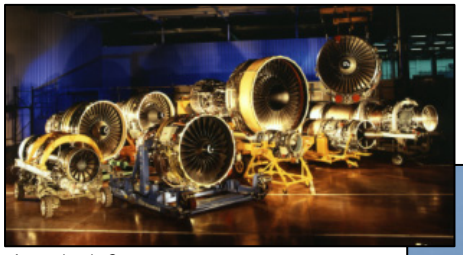
- Flight control systems
- Power management
- Reconfiguration management
- Autopilots
- Engine control systems (FADEC)
- Braking systems
- Fuel management
- Cockpit display and alarm management

EUROCOPTER – EC145

Dassault Aviation - Rafale

Dassault Aviation – Falcon 7X

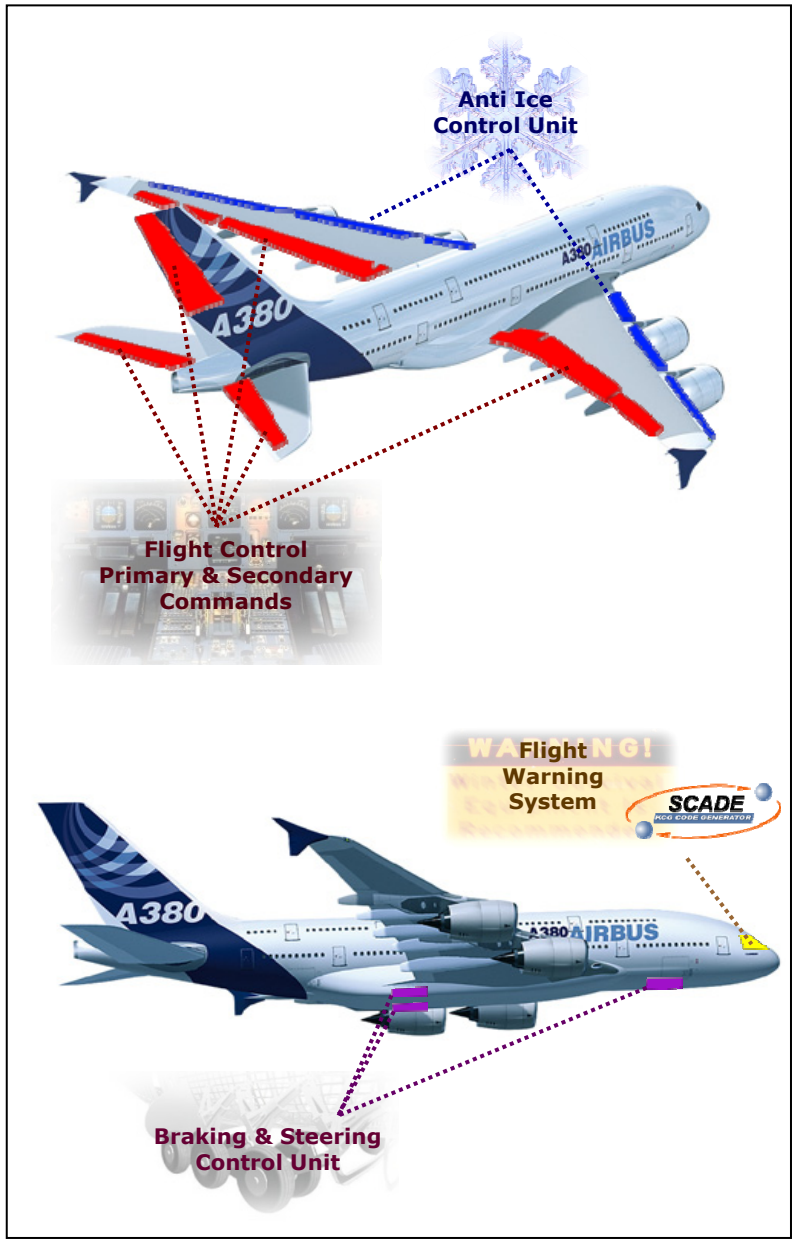
AIRBUS – A340-600 & A380



Aeroengines by Snecma
©Snecma/Studio Pons

US Air Force - F16

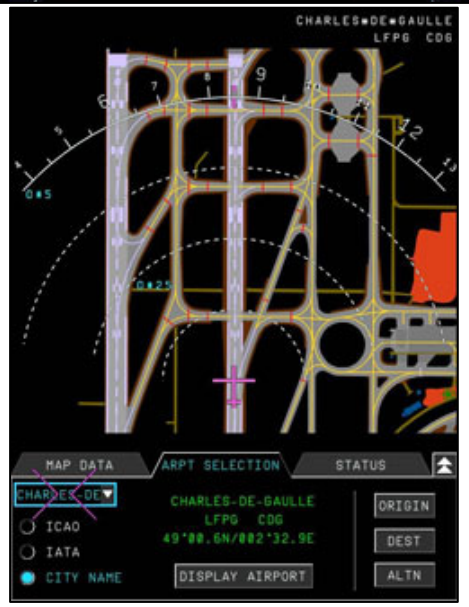
SCADE in the Airbus A380



- 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 in the A380 Cockpit

- Control and Display System (CDS)
  - Eight screens, two keyboards/cursor control devices
- Head-Up Display (HUD)
  - Incorporating LCD technology
- On-board Airport Navigation System (OANS)
  - SCADE Display & OpenGL graphics



SCADE in the Railways

- Interlocking systems control
- Signaling
- Ground stations
- Automatic Train Operations
- Train Control Systems
- Critical Graphics Displays
- Level Crossings
- Safe Platforms

(EN 50128 Certified by TÜV – up to SIL 4)

Scade in Automotive & Industrial Applications

- Automotive & 2-Wheelers:
  - Airbags
  - Braking Systems, ABS & ESP
  - Steering
  - Chassis & Suspension Systems
  - Restraining systems
  - Engine regulation
  - X-By-Wire applications

- Heavy Duty Industrial systems:
  - Cranes
  - Tractors
  - Tanks
  - Earth Moving Machines
  - Trucks
  - Construction equipment
  - Mining machines, etc…

(IEC 61508 Certified by TÜV – up to SIL3)

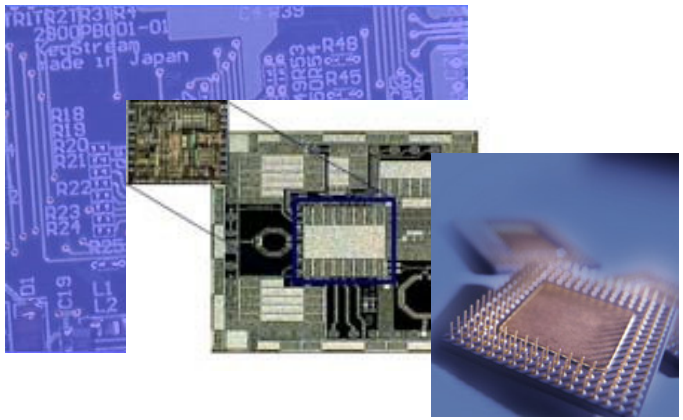
Esterel EDA Technologies & Esterel Consortium

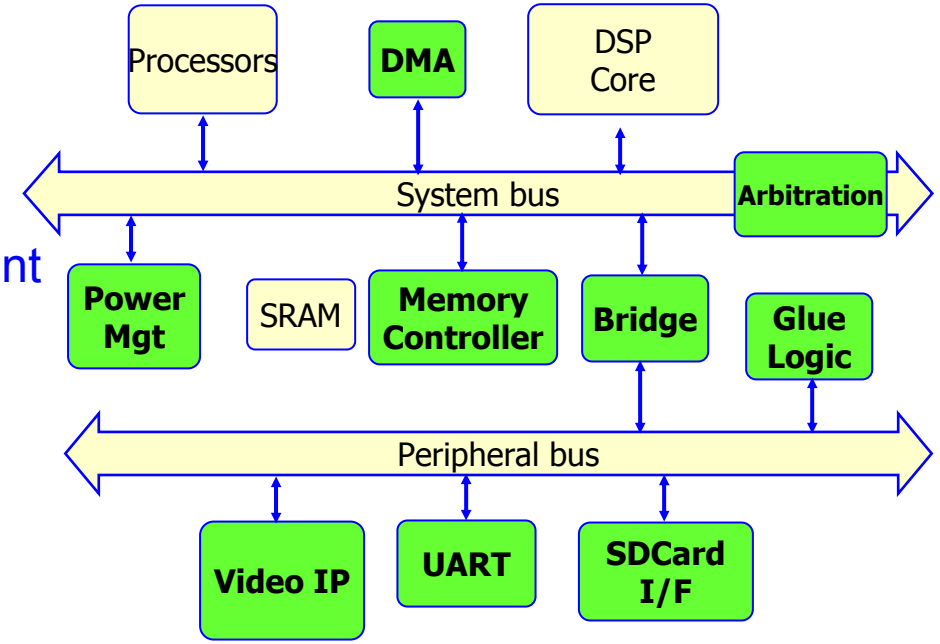
In 2001 Esterel Technologies formed a consortium of leading Semiconductor companies

- Early adopters of Esterel Studio™
- Best practice sharing about project use and design flow integration
- Collaborative specification of the main product features and roadmap
- Attended by academic partners for scientific advise
- Attended by Esterel Studio offer partners
- Support to the IEEE standardization process of the Esterel language

Esterel Application Targets

- Processor modeling and synthesis
  - Instruction Set Architecture
  - Complex instruction and data cache
  - Arbiters
  - Interrupt control
- Bus interfaces and peripheral controllers
  - Bus bridges, Networks on chips
  - Disk access, Serial ATA
  - Flash cards drivers
  - Video controllers
- Communication IPs
  - On-chip power and clock management
  - DMAs
  - Memory controllers
- Communication IPs
  - Protocols
  - Wireless links,
  - Fast serial links

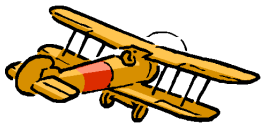




Beware of the computer!

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

Applications and Constraints

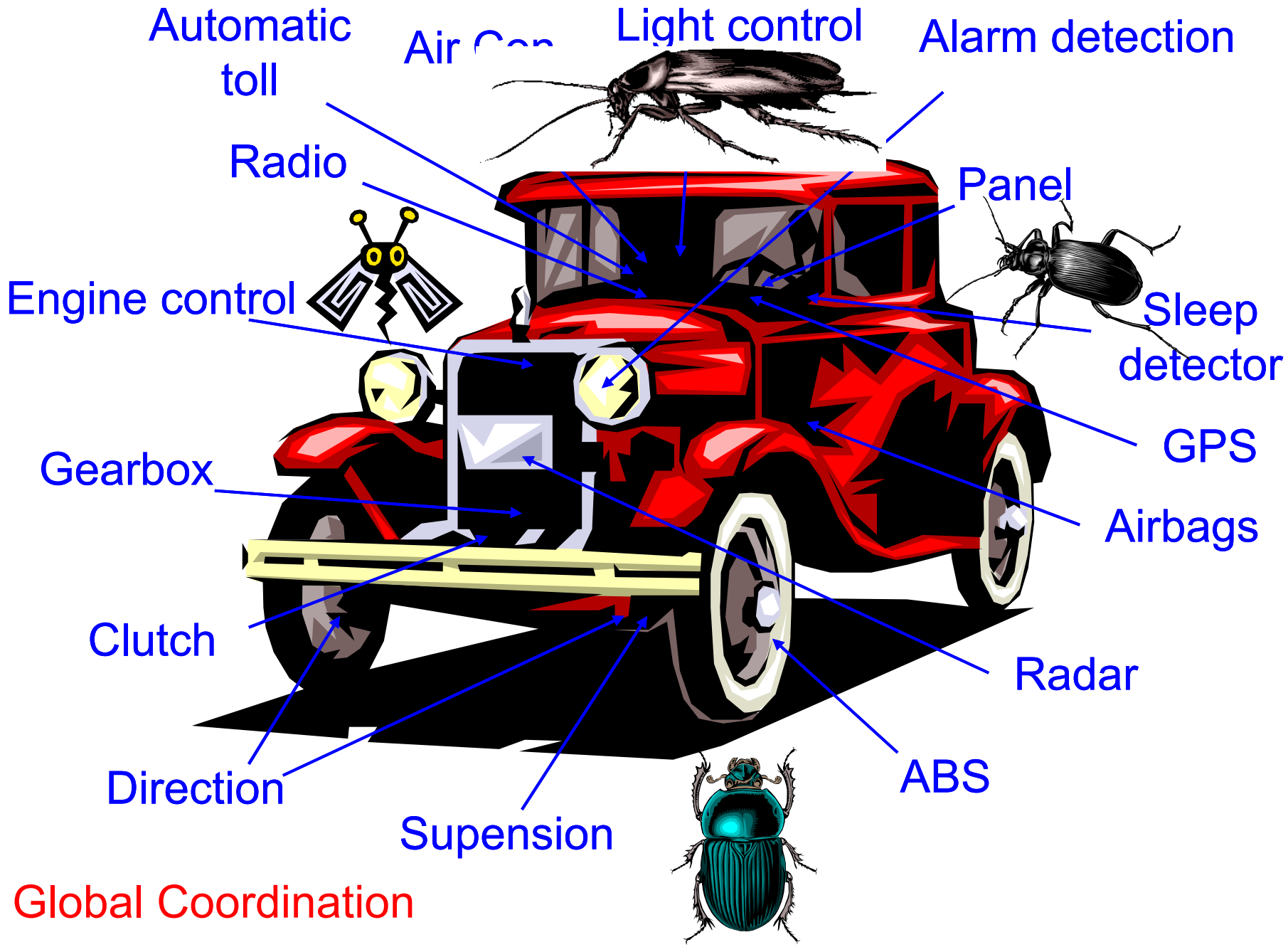


flight-control, engines, brakes, fuel, power, climate
safety-critical => certification

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

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

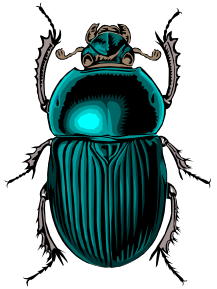
pacemakers, diabet control, robot surgeons
life-critical => TBD (I hope!)

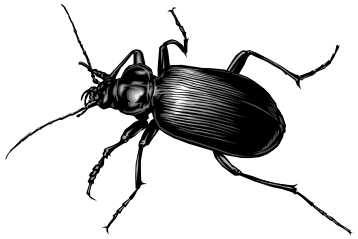


Enemy No 1 : the BUG

- Therac 25 : lethal irradiations
- Dharan's Patriot
- Ariane 501
- Mars satellites & Ro
- Automobile prob
- Intel & AMD
- Telepho amera bugs
- ...

Bug eradication campaign needed!



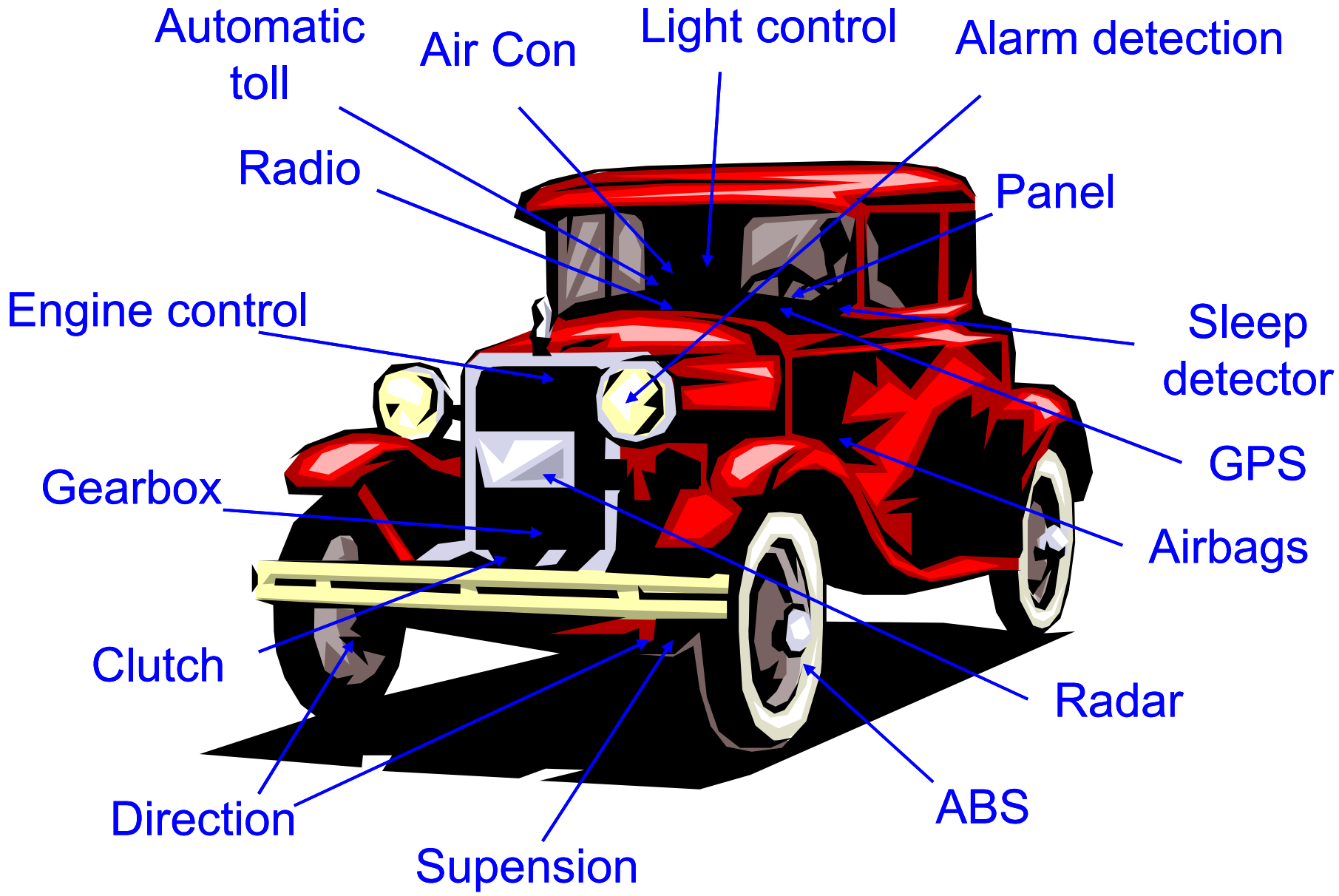


How to avoid or control bugs?

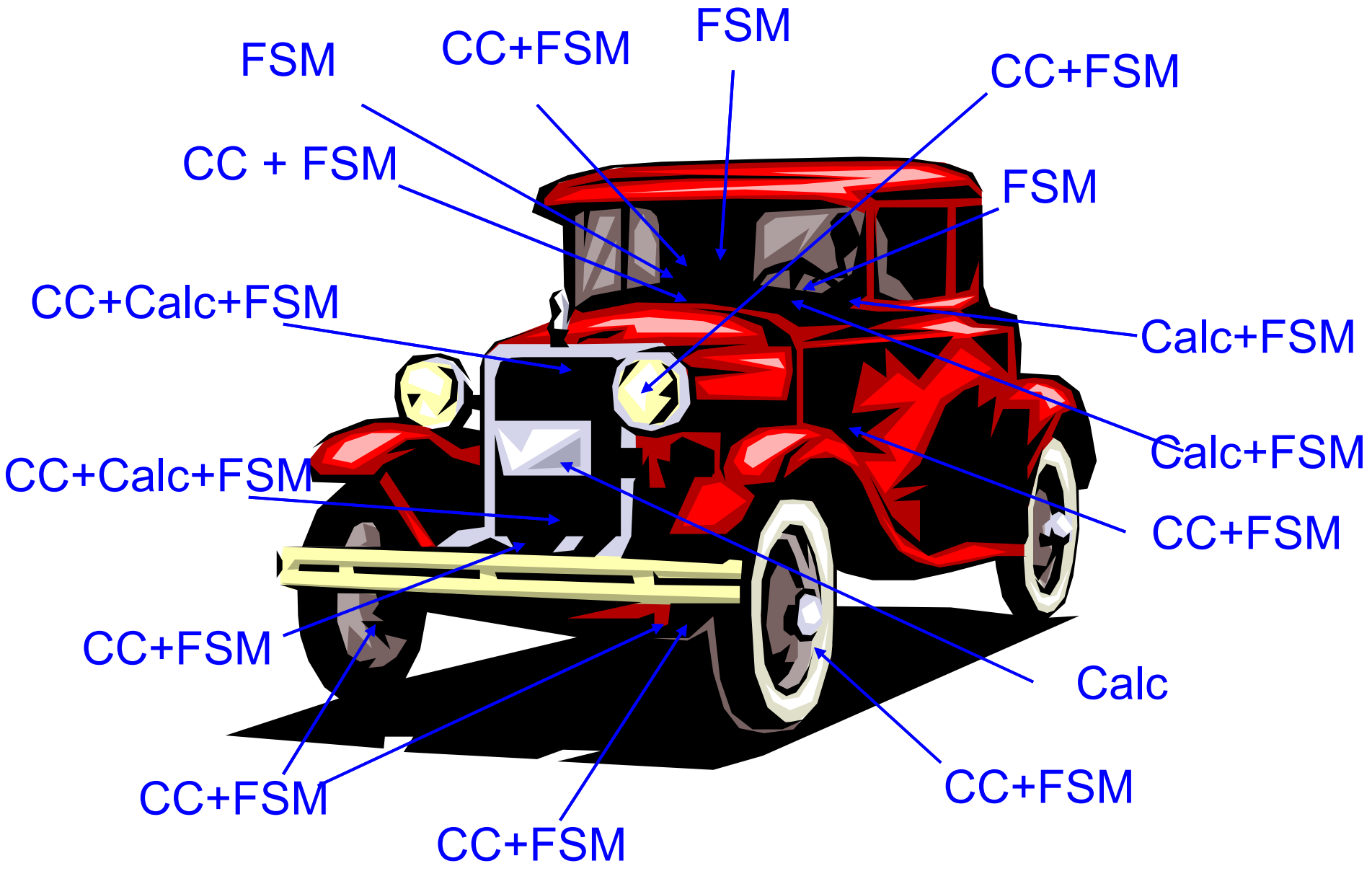
- Traditional : more verification by fancier simulation
  - but gets out of steam, more does not mean better
- Next step : better design
  - better and more reusable specifications
  - simpler computation models, formalisms, semantics
  - reduce architect / designer distance
  - reduce hardware / software distance
- Mandatory: better tooling
  - synthesis from high-level descriptions
  - formal property verification / program equivalence
  - certified libraries

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 / video
  - data flow networks, Java



Global Coordination



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.
  - but should never be allowed to go wild !
- Physical distribution becomes fundamental
  - separation of functions, links between them
  - redundancy for fault-tolerance
  - global time needed for distributed control

Bad News or Good News?

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)

The Classical Hardware Development Model is Inadequate

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

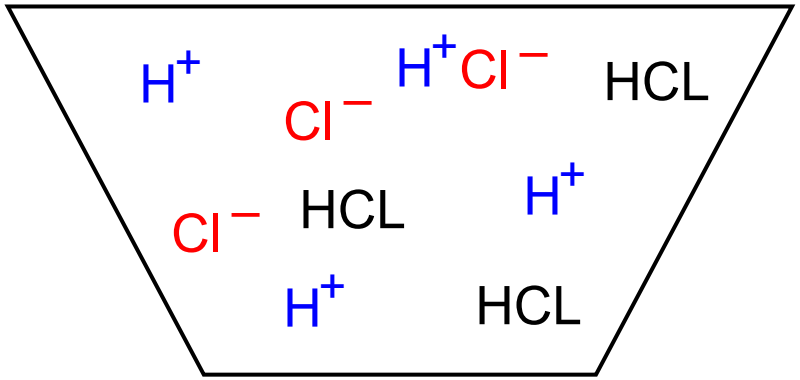
Other models are needed !

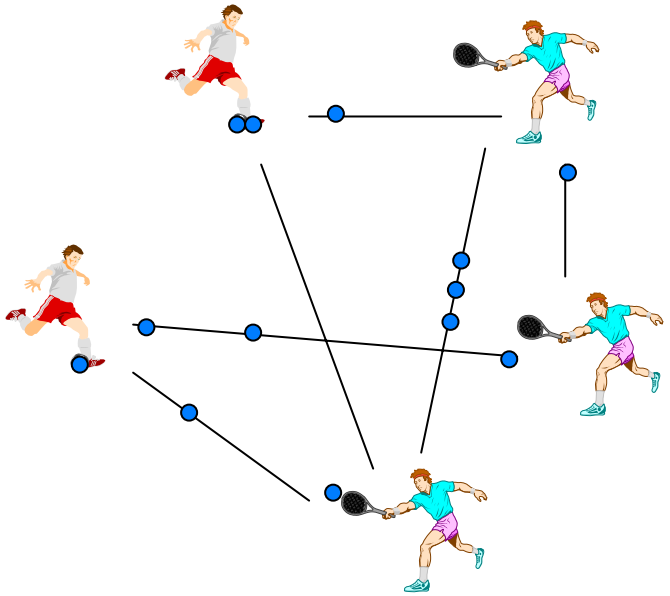
Concurrency models

Let *t* be the communication time
Then 3 fundamentally different models:

- *t* arbitrary — asynchrony
- *t* = 0 — synchrony
- *t* predictable — vibration

Arbitrary Delay : Brownian Motion





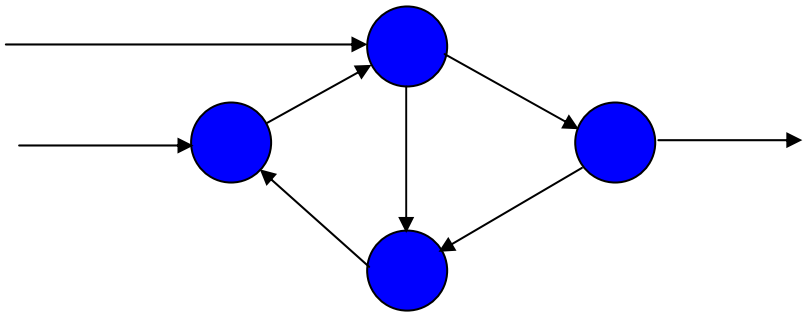
Chemical reaction

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

Internet routing

Models : Kahn networks, CSP / ADA,..., π-calculus, CHAM, Join-Calculus, Ambients,

Kahn Networks

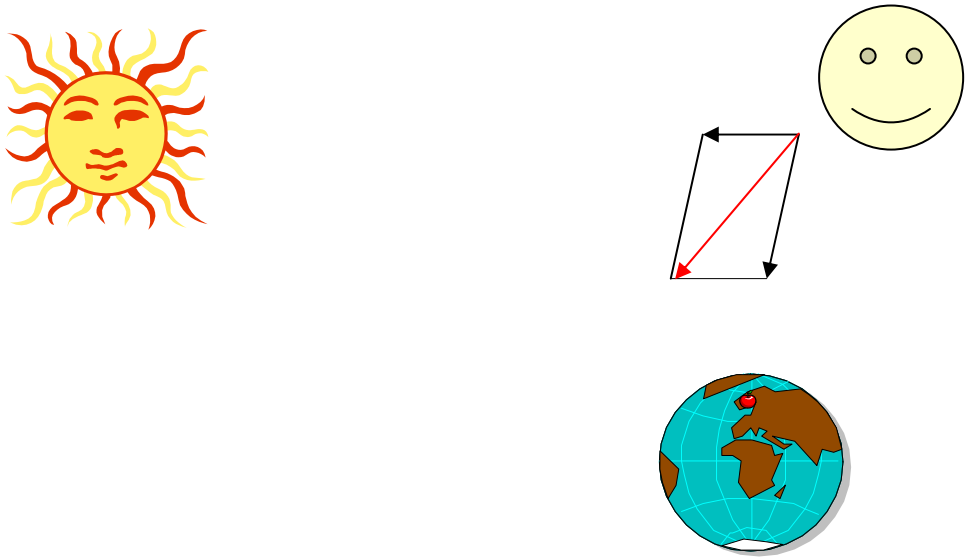


nodes = deterministic programs

arrows = infinite fifos

- result-deterministic (independent of computation order)
- easy semantics by flow equations
- heavily used in streaming applications (audio, TV)
- but semantics easily breaks down when language extended...

Zero delay: Newtonian Mechanics



Concurrency + Determinism
Calculations are feasible

Predictable time = 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, sound, 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, Bianca, Walter, and listeners can neglect the speed of sound

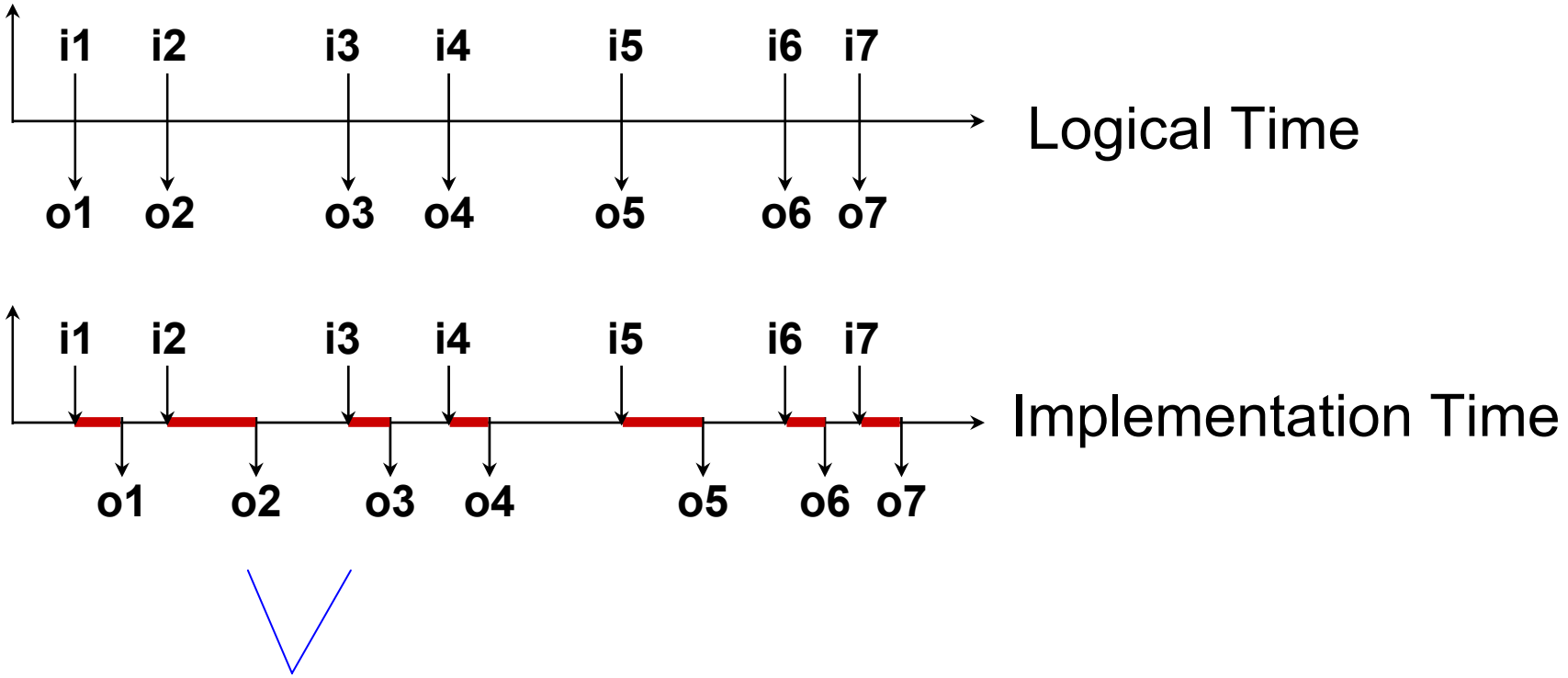
Specify with zero-delay

Implement with predictable delay

Control room size

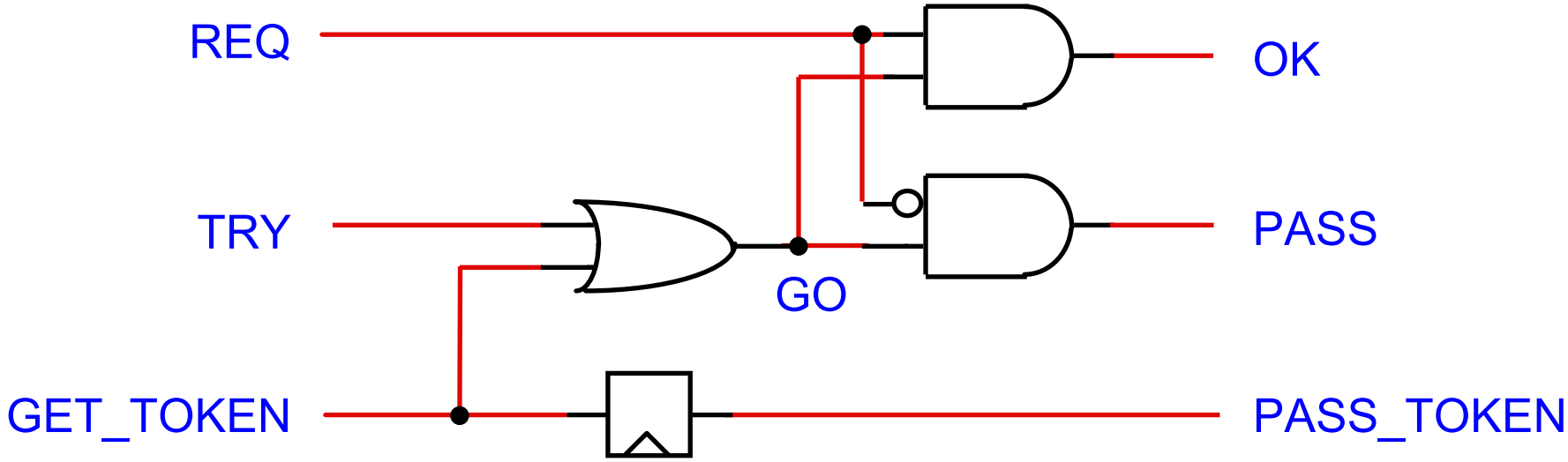
The Synchronous Models of Time

Time becomes a logical notion



WCET = guarantee of no-overlap

Hardware Synchrony: the RTL model

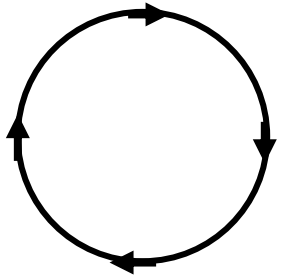


OK = REQ and GO
PASS = not REQ and GO
GO = TRY or GET_TOKEN
PASS_TOKEN = reg(GET_TOKEN)

Room size control = timing closure

Software Synchronous Systems

Cycle based



read inputs

compute reaction

produce outputs

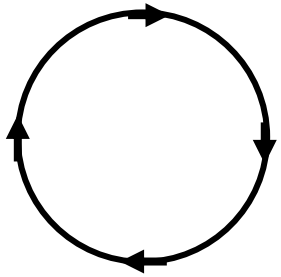
Synchronous = 0-delay = within the same cycle

propagate control

propagate signals

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

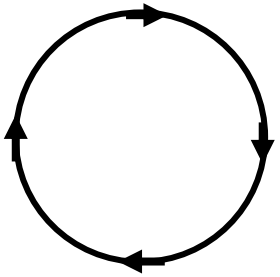
Concurrency = Cycle Fusion



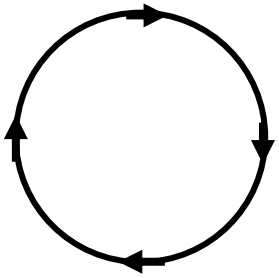
input X, Z;
output Y, T;
Y = X+1;
T=Z/2

Y

Z



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, makes WCET easier

A brief history of synchrony

- 1982-1985 : first ideas, languages, and semantics
  - Esterel : Berry – Marmorat - Rigault, Sophia-Antipolis
  - Lustre : Caspi – Halbwachs, Grenoble
  - Signal : Benveniste – Le Guernic, Rennes

  computer science
  control theory

- 1985-1998 : more languages, semantics, compiling & verification
  - SyncCharts (André), Reactive C (Boussinot), TCC (Saraswat), etc.
  - causality analysis (Berry, Gonthier, Shiple)
  - links to dataflow (Ptolemy), to hardware (Vuillemin), etc.
  - formal optimization & verification techniques (Madre & Coudert, Touati)
  - Creation of SCADE (IMAG, Verilog, Airbus, Schneider)

1991: extensive BDD-based formal verif. at Dassault Aviation

- 1998 –2008 : more research, maturation
  - S. Edwards, Synopsys
  - R.K. Shyamasundar, TIFR, S. Ramesh, IIT Mumbai
  - V. Saraswat, Xerox
  - K. Schneider, Karlsruhe / KaisersLautern : Quartz project
  - R. van Hanxleden, C. Traulsen, Kaiserslautern
  - L. Zaffalon, EIG Geneva

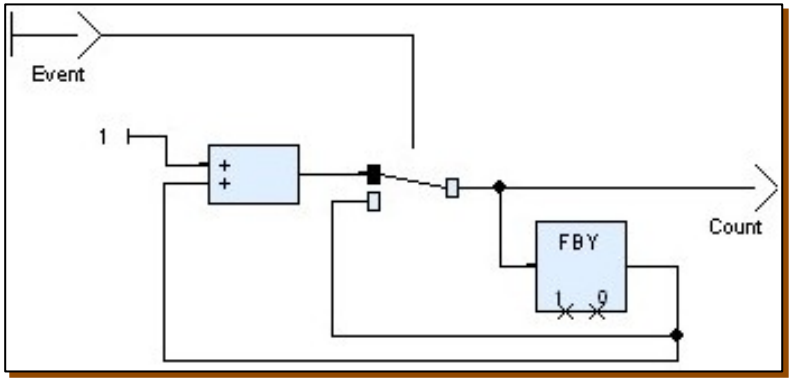
- 2001-2008 : industrial expansion
  - Development of Esterel v7 for hardware circuit design
  - Creation of the Esterel Consortium, IEEE Standardization of Esterel v7
  - Massive usage of SCADE in certified avionics embedded systems
  - Growing usage of SCADE in railways and automotive industries
  - Addition of SCADE Display

Esterel Studio and SCADE Suite are free for teaching activities

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))
```

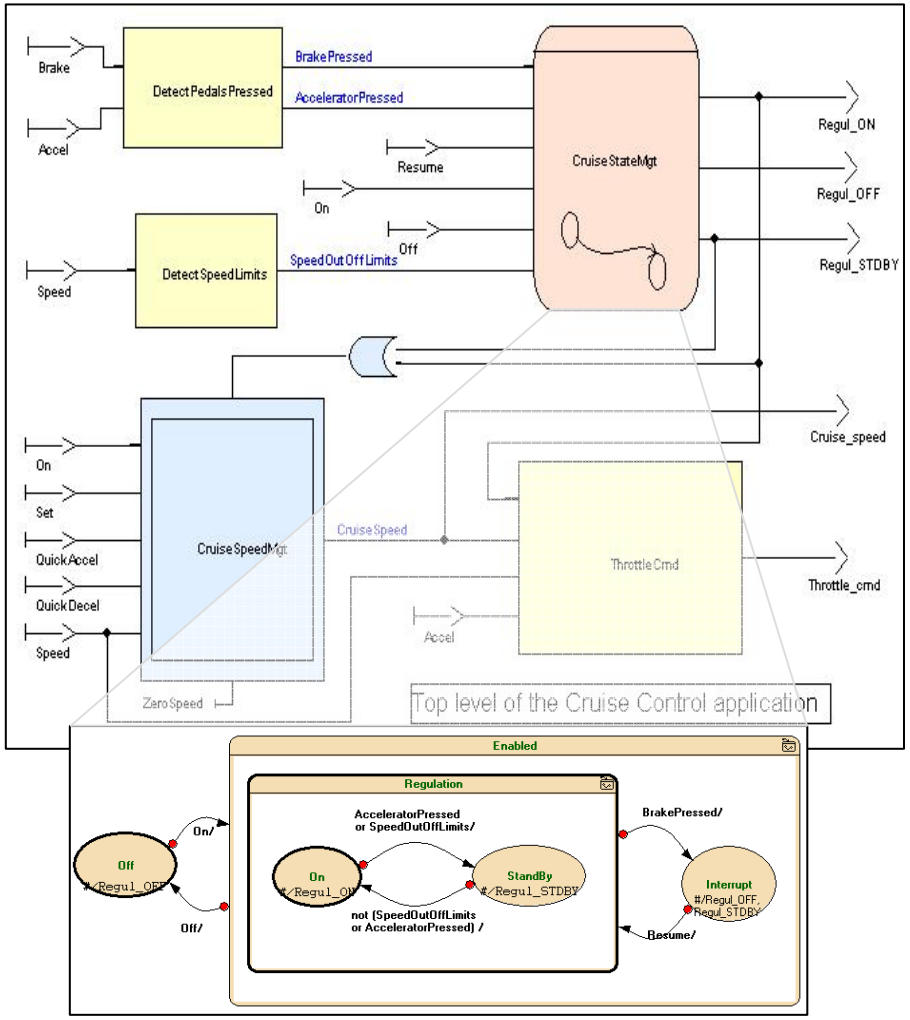
The **`Count`** flow is the solution of the equation

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 || CheckHeart
                    end every
                when 15 Second ;
                run FullSpeed
            each  Lap
        when  4 Lap
    end every
handle HeartAttack fo
   run RushToHospital
end trap
```

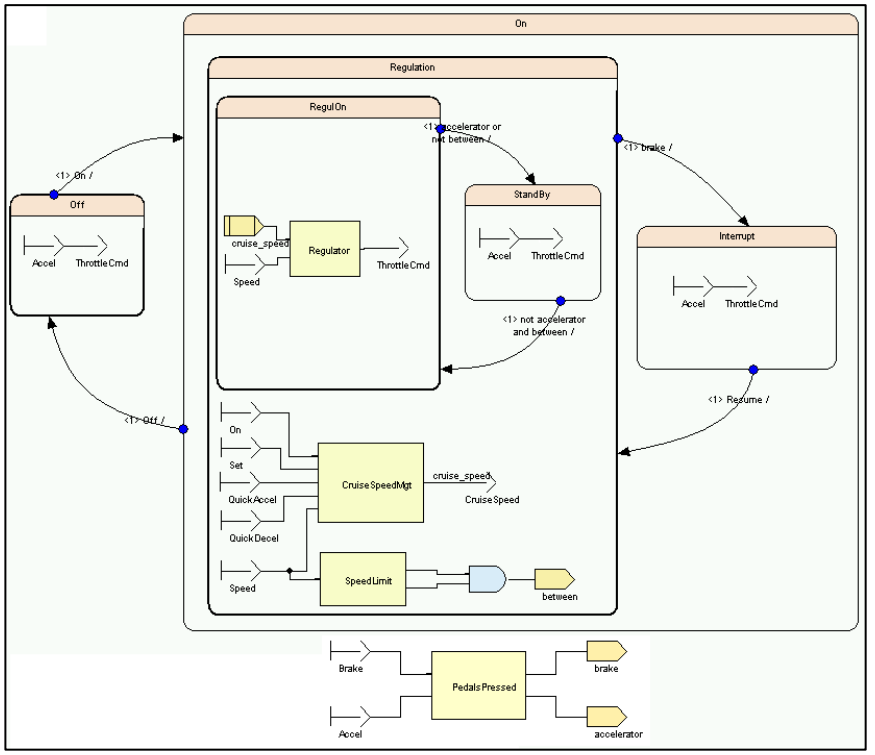

exit HeartAttack

Scade : data / control flow unification



SCADE 5

Pure FSM into Pure CC



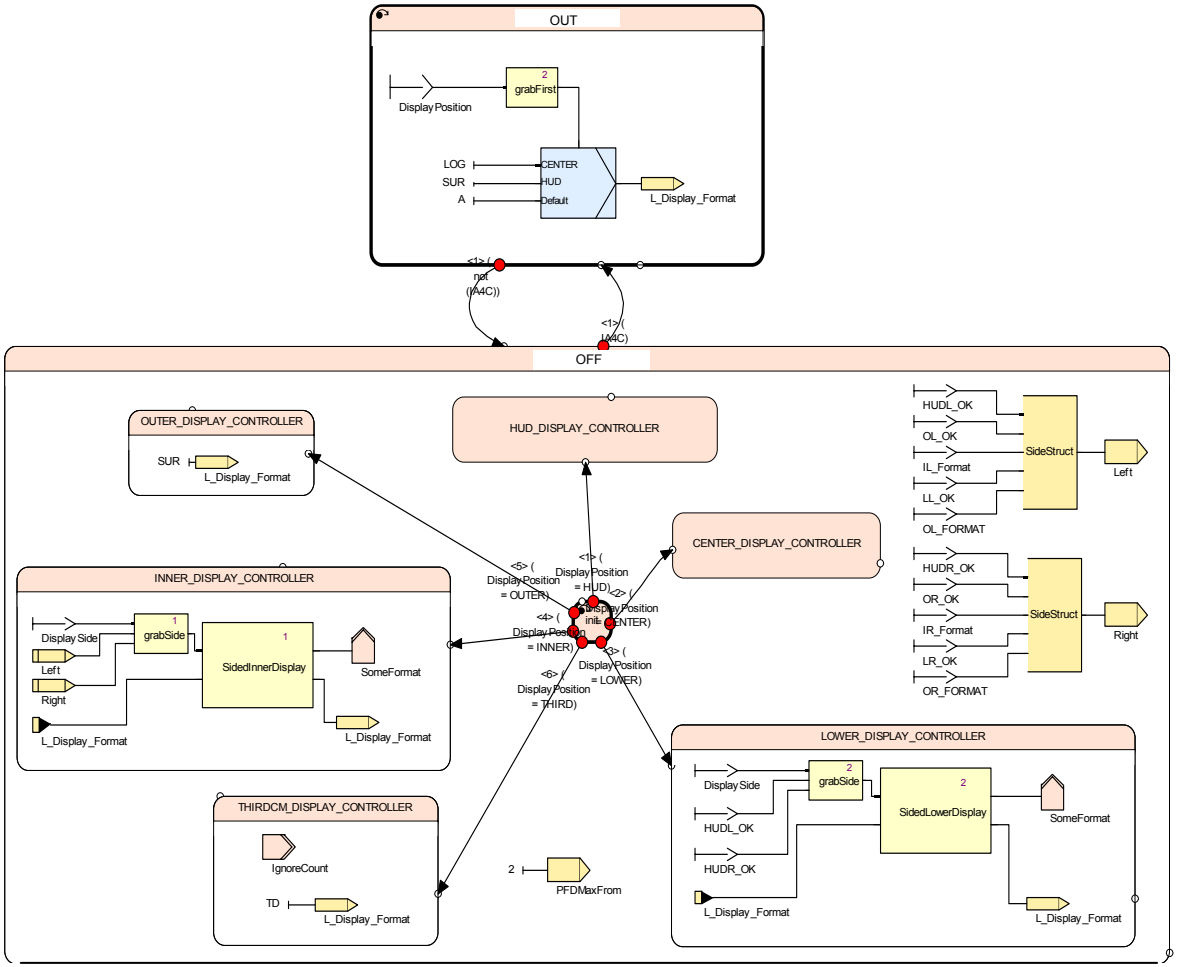
SCADE 6

Unified FSM & CC

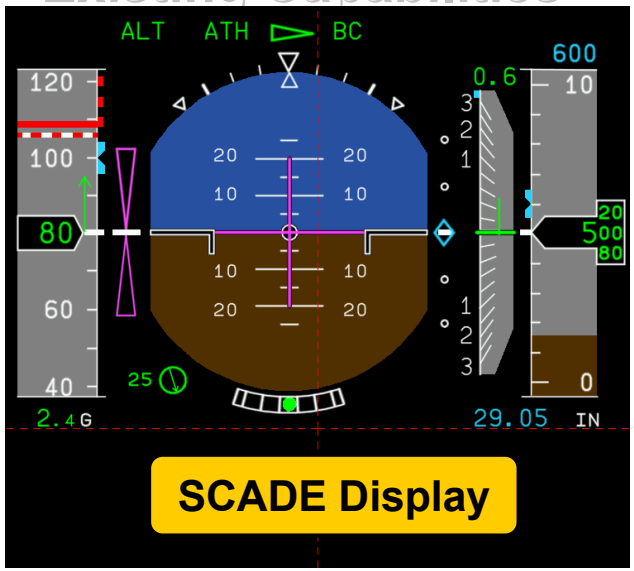
Freely mixable in hierarchy

Functional language,
Functional array support
Formal semantics

A Typical Application: Cockpit Display



Existing Capabilities



Design and Verification Flows

- Industrial develoment is about flows, not just tools
- Flow : full path from requirements to final object
- Methods and tools make sense only if integrated in official (non-R&D) production flows
- Verification is not a single activity but appears everywhere, and should be itself verified
- Flows cannot evolve fast

DO-178B certified avionics software flow

- Process based certification by independent authority (FAA, CEAT, JAA, etc.), use worldwide since 1992
- Goal: detect and report errors introduced during software development
- Verification objectives defined, but no specific development / verification techniques promoted
- Verification is not just testing. It contains also reviews and deep traceability-based analyses of the entire process
- Verification of verification is mandatory

Special nature of software is acknowledged

Towards DO-178C

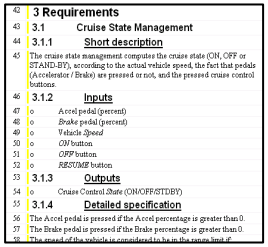
- 2005-2009 : Working Group, 120 people, 1000 on Web site
- From process-based to product-based
  - tool qualification
  - model-based development
  - OO design
  - automatic code generation
  - formal verification
- Full consensus needed to publish the document

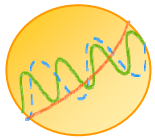
Other Standards

- DO-254: avionics hardware development
- IEC 61508: function safety of systems made with Electrical, Electronic, Programmable electronic components
- EN 50128: Adaptation of IEC 61508 to Railways
- MIL-STD-498: Military standard for SW development
- DEF-STD-055/056: Safety management for Defense Systems
- Chinese Standards

The SCADE™ Certified Software Factory

SYSTEM SPEC

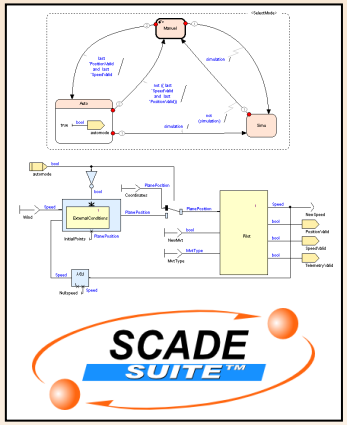


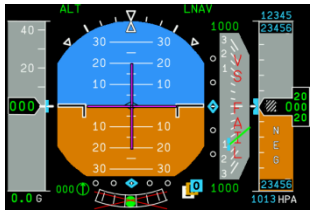


Algorithm Design Capture

Architecture Design Capture

DESIGN





VERIFY

Debugging & Simulation

Formal Verification

Model Coverage Analysis

Object Code Verification

SCADE Suite/SCADE Display Integration

Graphical Animation

Ergonomics Checking

GENERATE

SCADE Suite KCG

RTOS Wrappers

SCADE Display KCG

OpenGL SC Compliant

SYSTEM TEST

MANAGE & TRACE

Requirements Management Gateway

Integrated Configuration Management

Automatic Design Documentation

DO-178B
IEC 61508
EN 50128

Qualification Kits, Certificates & Handbooks

Code Generation with KCG

- KCG is the qualifiable C code generator
  - developed in CAML with a DO-178B Level A process
  - certification authorities certified that “KCG can fly” and qualified it as a development tool

  => no need to unit-test the generated C code
- Evidences provided to users
  - qualification kit
  - verifiable, traceable, and safe code
  - C compiler verification kit

Synchronous Semantics

- Ensures every data is produced exactly once
- Additional static checks
  - no access to undefined data
  - no race condition (combinational cycle)
    => deterministic scheduling-independent result
  - no recursion in node calls
    => static memory allocation, bounded stack

Checked by the qualified code generator as a prerequisite to code generation

Generated Code Properties

- Small C subset
- Portable (compiler, target and OS independent)
- Structured (by function or by blocks)
- Readable, traceable (names/annotations propagation)
- Safe static memory allocation
- No pointer arithmetic
- No recursion, no loop
- Bounded execution time
- Size and / or speed optimizations

Eases verification and static analysis (Astrée, AbsInt)

Agenda

- About Esterel Technologies
- Beware of the Computer!
- Design and Verification Flows for Embedded SW
- Design and Verification Flows for SoCs
- The Synchronous Approach to D&V
- Overview of SCADE
- Overview of Esterel Studio

System-on-Chips Flows

- Entirely in-house
- Long chain of individually hard flow components
  - informal documentation (English)
  - manual coding at RTL level (VHDL, Verilog)
  - semi-automatic design for testability (DFT) additions
  - automatic logic synthesis
  - automatic place and route
  - mask fabrication
  - final chip on-line testing
- A hard milestone : RTL sign-off
  - after that, mask patches needed, 100,000$ +

Architecture

components
dimensioning
communication

Word, Excel, Visio
System C

Micro-Architecture

concurrency
pipeline
resource sharing

Word, Visio, C

RTL design

gates, clocks
registers, RAMs
critical path

VHDL, Verilog

circuits

cells, clock trees
area, speed

netlists

DFT (test)

testability
scan insertion

netlists

Place&Route

physical & electrical
constraints

P&R netlists

$ 1,000,000 Masks

printing

pseudo-rectangles

Chips

fabrication

silicon dies

Architecture

functionality OK?
throughput OK?
marketing OK?

Experience
Reviews

↓

Micro-Architecture

breakdown OK?
performance OK?

C-based modeling

↓

RTL design

functionality OK?
area/speed OK?
power OK?

Random-directed testing,
Formal verification

↓

circuits

equivalent to RTL?

formal equivalence checking

↓

DFT (test)

test coverage ~100% ?

ATPG

↓

Place&Route

connections?
electrical constraints?
timing?

Design Rules Checking (DRC)

↓

$ 1,000,000 Masks

No fab fault?

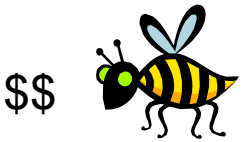
Scan test run

↓

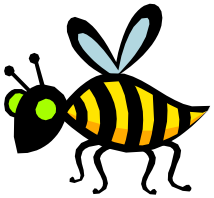
Chips

Architecture

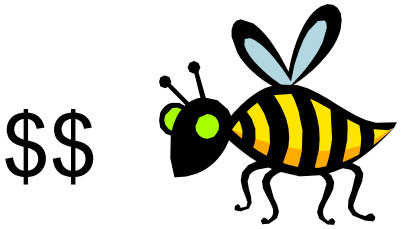
Micro-Architecture



RTL design



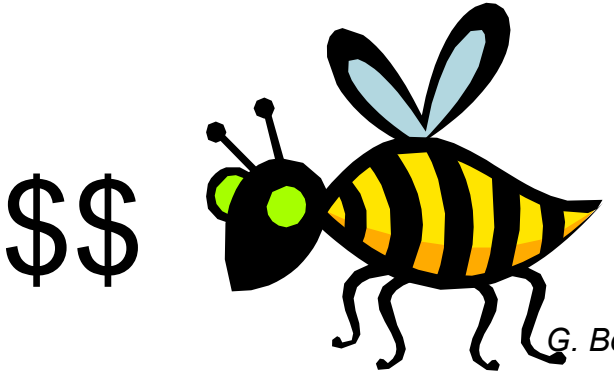
circuits



DFT (test)

Place&Route

$ 1,000,000 Masks



Chips

Esterel

Architecture

Micro-Architecture

RTL design

circuits

DFT (test)

Place&Route

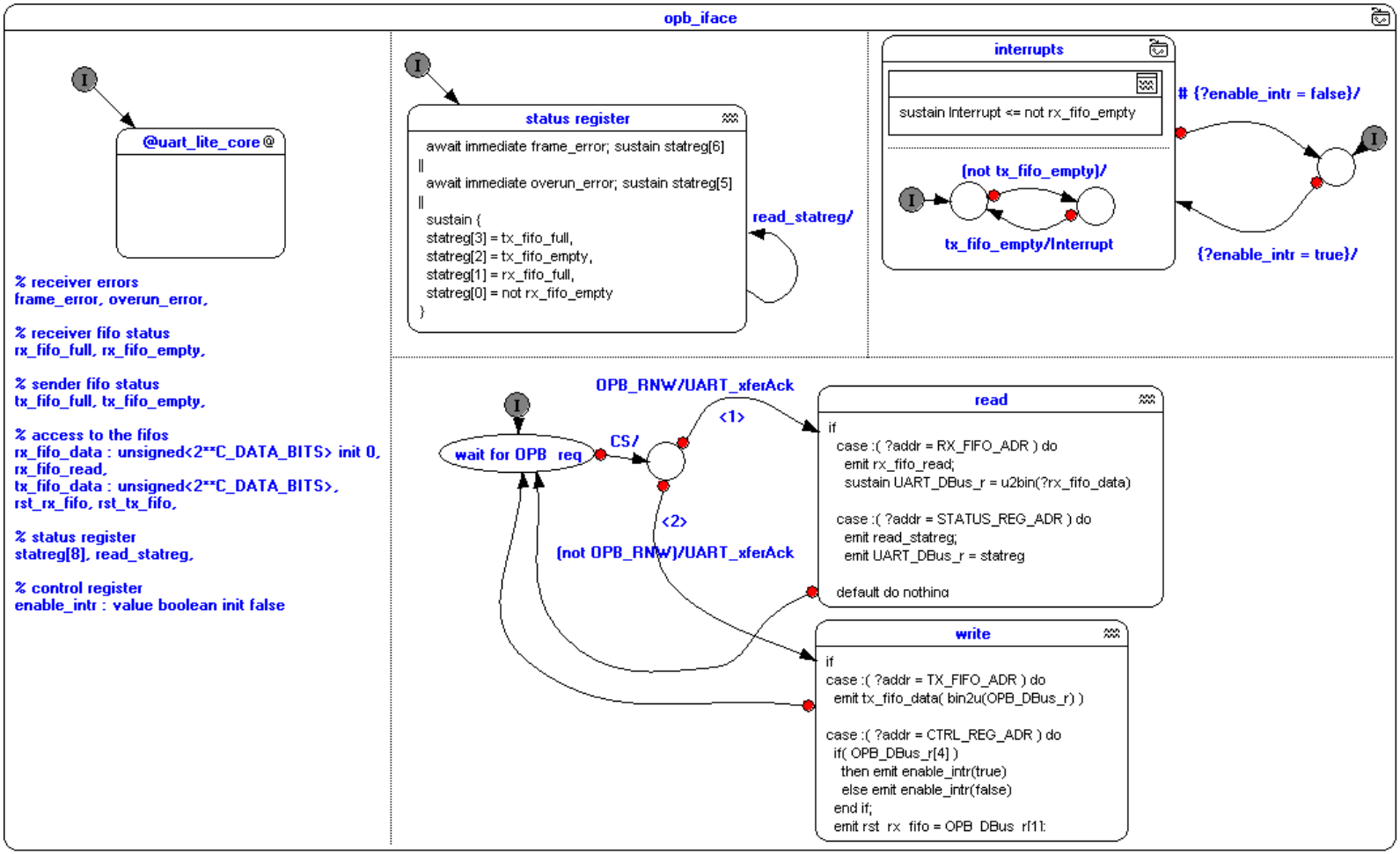
$ 1,000,000 Masks

Chips

Key Messages to Users

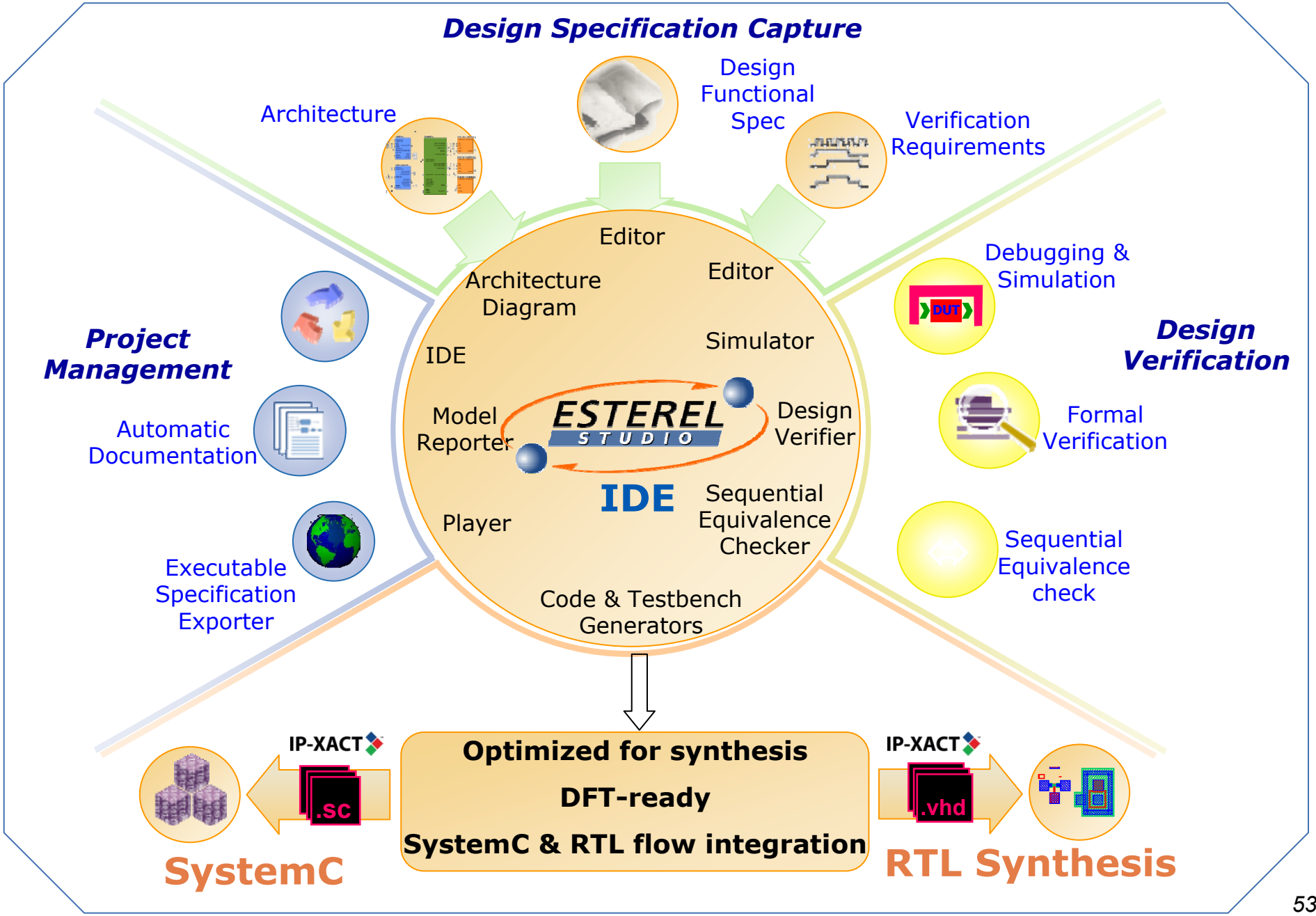
1. Specification of dynamics cannot be accurate when written on static paper
2. Animated executable specifications key to reuse, inter-teams communication, what-if studies, etc
3. Once such specs are available, why recoding?
   - single model for HW synthesis and SW modeling (SystemC)
4. Spec-to-implementation path : formal methods and tools
   - hierarchical behavior description
   - languages with formal semantics
   - formal compiling algorithms
   - formal verification techniques
5. Formal verification= design tool usable at all design steps

Esterel v7 (Berry – Kishinevsky)



text + graphics, concurrency + sequencing
clear semantics

Esterel Studio at a glance



From rooms to castles (GALS): reconciling synchrony and asynchrony

- Time-triggered local area networks : TTP, FlexRay, etc.
  - built-in determinism and fault tolerance
  - based on clock synchronization
- Mutual sampling of computers
  - based on distributed Nyquist theorem (Caspi & Benveniste)
  - works because of control theory, not just computer science
- Multiclock circuits
  - multiple clock zones for processor, DSP, accelerators, etc.
  - beware of metastability issues!

Clever research results still needed!

Conclusion

- Synchronous formal methods are heavily used in industry
  - formal languages
  - formal compilation schemes
  - formal verification
- They make verification much simpler
  - Source language matters, behavior hierarchy is key
- Current research & development
  - Improving the languages (SCADE 6, IEEE-standard Esterel)
  - Improving the compilers (faster and more modular output)
  - Scaling up formal verification : how big can you verify?

Get Esterel Studio and SCADE free
for teaching and academic usage