

LOW COST HIGH INTEGRITY PLATFORM



Thierry Lecomte

David Deharbe

Patrick Péronne

Etienne Prun

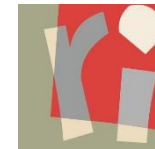
Denis Sabatier



Emmanuel Chailloux

Adilla Susungi

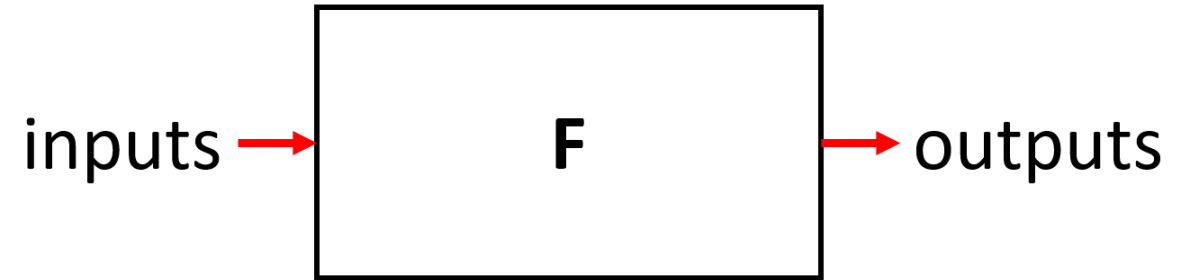
Steven Varoumas



Sylvain Conchon

What it is all about ?

Safety computer



- $F == (\text{read inputs, compute, set outputs})^*$
- F could harm / kill people
- Ability to check if able to execute F properly

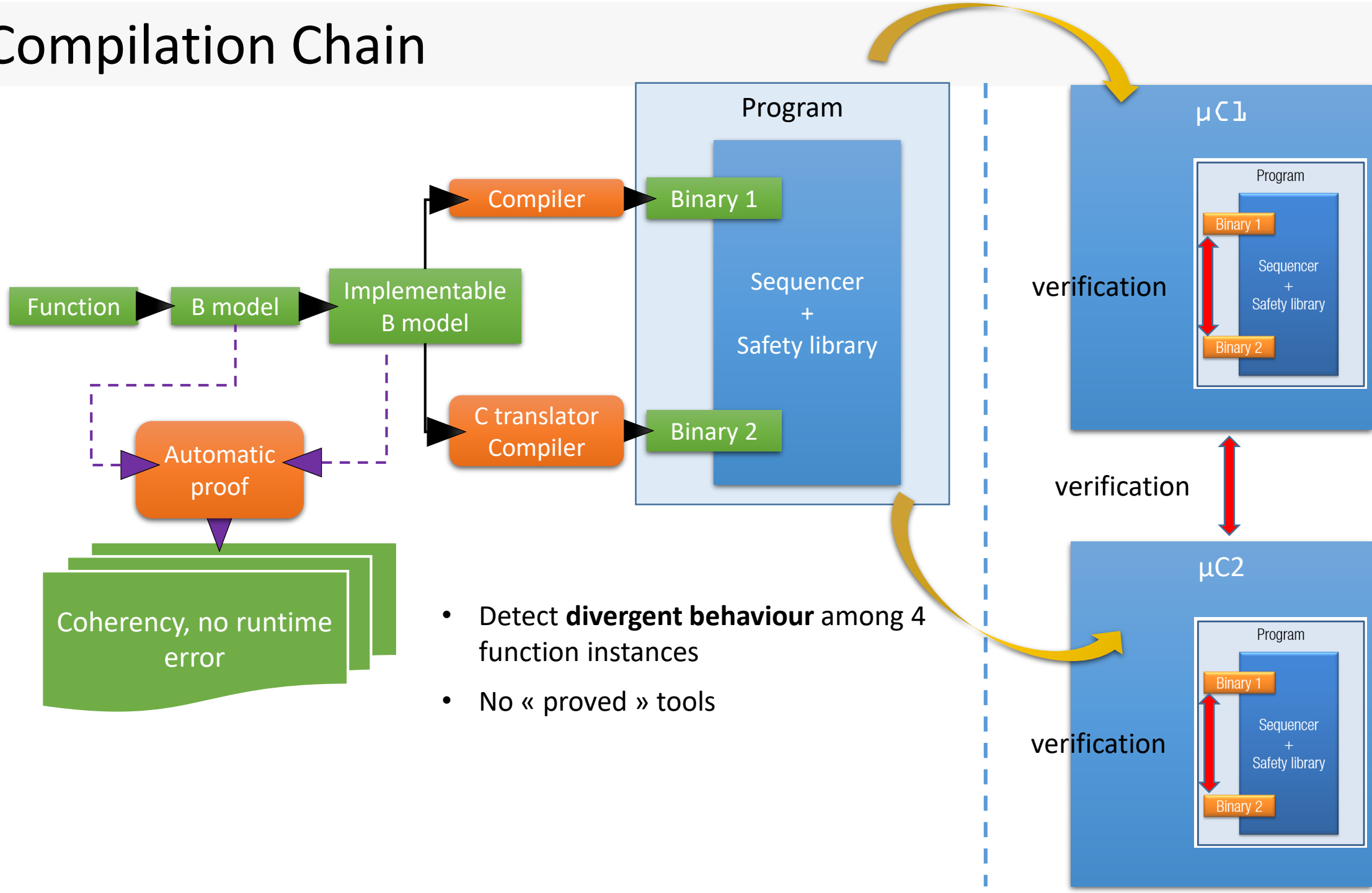


- F is not safe just because a safety computer is used
« execute the right F and execute the F right »

Principles

- ▶ Formal model could be obtained from a DSL
 - ▶ F **proved formal model** (validation testing only)
 - ▶ F code generated from model
 - ▶ Proof and code generation (fully) automatic
-
- ▶ 2 identical microcontrollers to execute several diverse instances of F
 - ▶ Continuous behavioral verification


Double Compilation Chain



Verification

Safety is built-in, out of reach of the developer who cannot alter it

If one verification fails when loading or executing

- 
- Bad CRC when bootloading code
 - Bad memory map (overlap) when bootload
 - $\text{CRC}(\text{data}_{\text{Binary1}}) \neq \text{CRC}(\text{data}_{\text{Binary2}})$ during execution on one μC
 - Failing μC unable to handshake every 50 ms with other μC
 - $\text{CRC}(\text{data}_{\text{Binary}})$ different on each μC (inter μC verification)
 - Wrong input (absence of/incorrect sinusoidal signal)
 - Outputs are not commandable
 - **Output is ON when both μC agree**
 - One μC is not able to execute properly instructions
 - $\text{CRC}_{\text{computed}}(\text{code}) \neq \text{CRC}_{\text{expected}}(\text{code})$ (deferred action)
 - Etc.

Handle failures:

- **Systematic** (buggy code generator, etc.)
- **Random** (memory corruption, failing transistor, degrading clock, etc.)

Models are proved to be correct:

- Syntax, types, properties
- No overflow, no division by 0, no access to a table outside of its bounds

Hyp

Programming Model & Applications

- ▶ The execution is cyclic
- ▶ The function is executed regularly as often as possible similar to arduino programming (setup(), loop())
- ▶ No underlying operating system
- ▶ No interrupt()
- ▶ No predefined cycle time (if outputs are not set and cross read every **50ms**, board enters panic mode)
- ▶ No delay()
- ▶ Inputs are values captured at the beginning of a cycle (digital I/O)
- ▶ Outputs are maintained from one cycle to another (digital I/O)
- ▶ Project skeleton is generated from board description (I/O used, naming)
- ▶ Programming is specifying and implementing the function *user_logic*

```
init();  
  
while (1) {  
    instance1();  
    instance2();  
}
```

Example (specification)



- ▶ I1, I2, O1, O2 belongs to `{IO_OFF, IO_ON}` • is unsigned 8 bit integer enumeration
- ▶ O1 is IO_ON *iff* both I1 and I2 are IO_ON
- ▶ O2 is the complement of O1

```
user_logic =
```

```
BEGIN
```

```
    O1, O2: (  
        O1 : uint8_t &  
        O2 : uint8_t  
    )
```

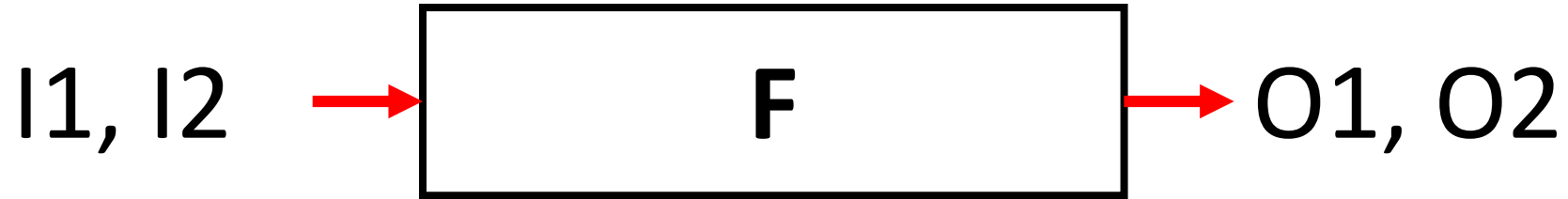
```
END;
```

« O1, O2 become such that

- Type of O1 is unsigned 8 bit integer
- Type of O2 is unsigned 8 bit integer »

Minimum specification :« O1 and O2 are modified in accordance with their type »

Example (specification)



- ▶ I1, I2, O1, O2 belongs to {IO_OFF, IO_ON}
- ▶ O1 is IO_ON *iff* both I1 and I2 are IO_ON
- ▶ O2 is the complement of O1

```
user_logic =
```

```
BEGIN
```

```
    O1, O2: (
```

```
        O1 : uint8_t &
```

```
        O2 : uint8_t &
```

```
        (O1=IO_ON <=> (I1=IO_ON & I2=IO_ON)) &
```

```
        not (O1 = O2)
```

```
    )
```

```
END
```


Example (implementation)



Green means
« Implementation
Fully proved
Against
Specification »

1/1
1/1

```
user_logic =  
BEGIN  
  VAR i1_, i2_ IN  
    i1_ : (i1_ : uint8_t);  
    i2_ : (i2_ : uint8_t);  
  
    i1_ <-- get_I1;  
    i2_ <-- get_I2;  
  
    O1 := IO_OFF;  
    O2 := IO_ON;  
    IF i1_ = IO_ON THEN  
      IF i2_ = IO_ON THEN  
        O1 := IO_ON;  
        O2 := IO_OFF;  
      END  
    END  
  END  
END  
END
```

Get I1 and I2 values

Example (code generation)

user_logic =

BEGIN

VAR i1_, i2_ IN

i1_ : (i1_ : uint8_t);
i2_ : (i2_ : uint8_t);

i1_ <-- get_I1;
i2_ <-- get_I2;

O1 := IO_OFF;
O2 := IO_ON;
IF i1_ = IO_ON THEN
 IF i2_ = IO_ON THEN
 O1 := IO_ON;
 O2 := IO_OFF;
 END
END

END

END

~~void~~ SECTION_C4B_FUNCTION user_logic(void)

{

{

uint8_t i1_;
uint8_t i2_;

get_I1(&i1_);
get_I2(&i2_);

O1 = IO_OFF;
O2 = IO_ON;
if(i1_ == IO_ON)
{
 if(i2_ == IO_ON)
 {
 O1 = IO_ON;
 O2 = IO_OFF;
 }
}

}

}

Initially Developed during a R&D Project [FUI21]

Consortium

CLEARSY
SYSTEMS ENGINEERING



OCaml PRO



Support



Ease the development of safety-critical applications

Safety built-in, formal method, dedicated hardware

► Includes

- Development of hardware & software generic platform
- Strengthening the automatic proof capability
- **Connection to domain specific languages**
- Automatic binary code generating and boot loading

► Reuse existing building blocks developed for railways

► Automate the development process

► Ease the certification process

► Obtain generic version to address other domains

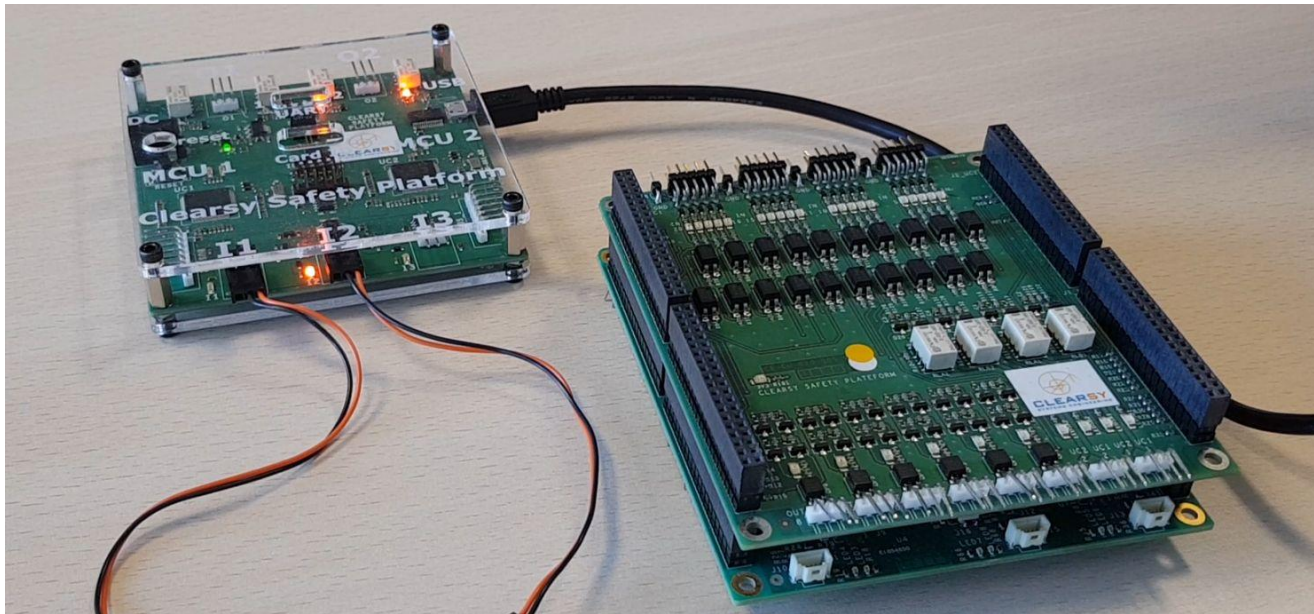
Funding

bpifrance



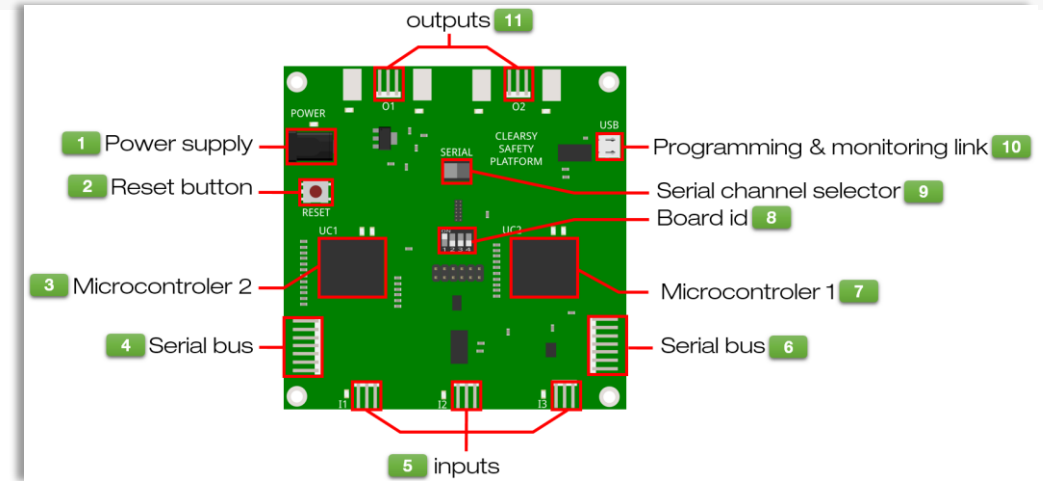
Dissemination and References

- **Starter kits for education:**
 - SK0 available since Q1 2019: 5 digital I/O
 - SK1 since Q2 2019: 26 digital I/O



Starter kits: SK0 (left) and SK1 (right)

<https://www.clearsy.com/en/our-tools/clearsy-safety-platform/>



- **Courses (up to Master 2):**
 - CentraleSupélec Paris (France)
 - LORIA Nancy (France)
 - Univ. Créteil / Paris (France)
 - Univ. Firenze (Italy)
 - Univ. Braga (Portugal)
 - UFRN Natal (Brazil)
 - UFF Rio (Brazil)
 - Univ. Sherbrooke (Canada)

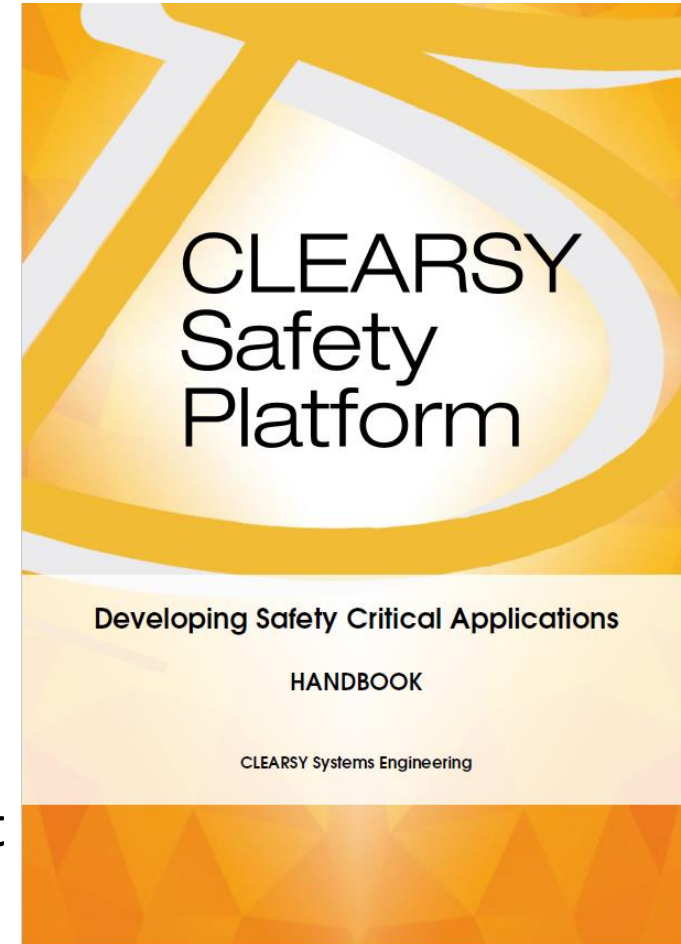
Dissemination and References



- 2 years of hands-on sessions and M2 courses in Europe, North and South America
- Feedback collected to improve the IDE and the hardware



- Free handbook for software development



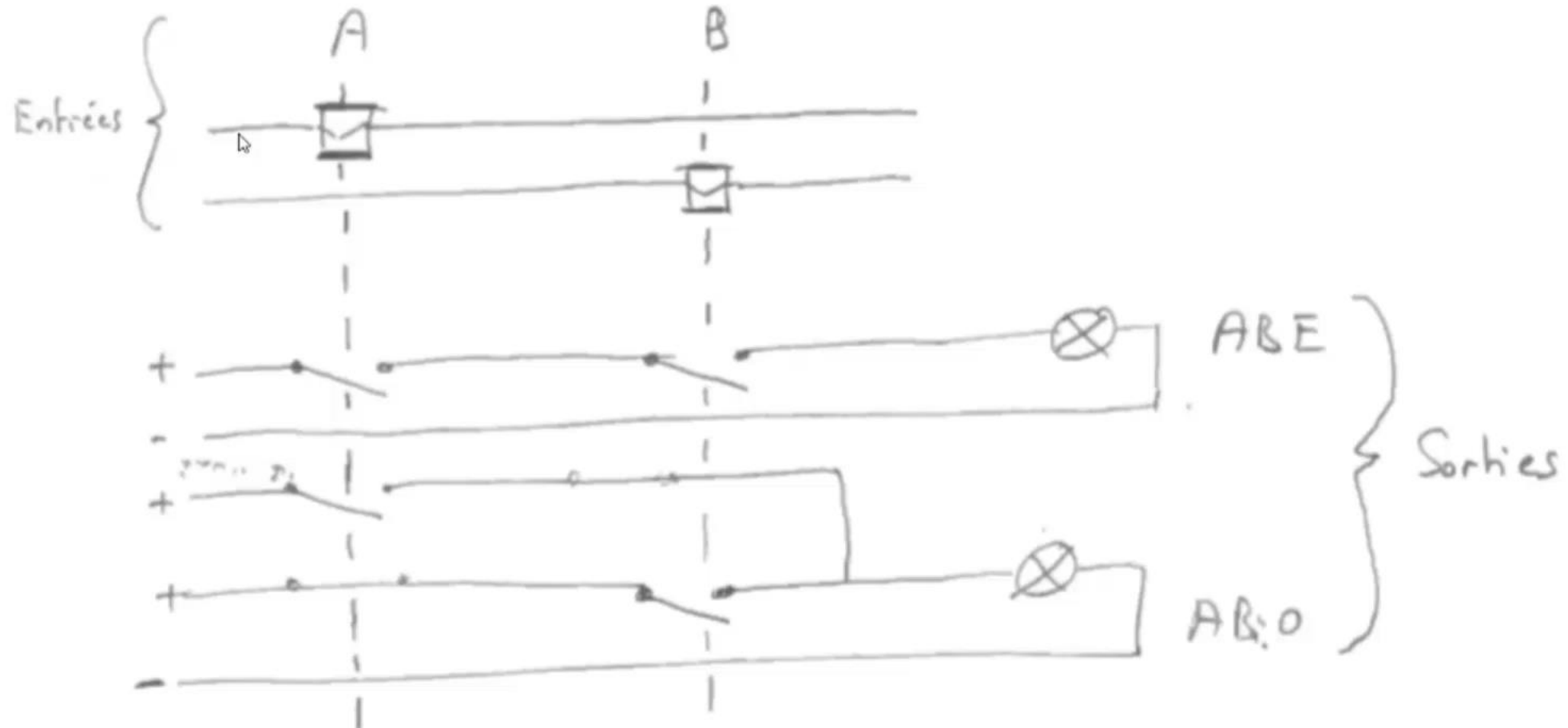
<https://www.clearsy.com/en/download/download-documentation/>

Connection with Domain Specific Languages

- ▶ Allow engineers to work with their usual development/modeling environment
- ▶ Formal framework behind the curtain
- ▶ **Certification-friendly**: B formal model could be handwritten or generated
- ▶ SNCF use cases: temporary work zone signaling, level crossing control
- ▶ Application test: relay-based schematic simple translation towards electronic platform (video)
- ▶ On-going research with RoboSim (model simulations of robotics systems)

Connection with Domain Specific Languages

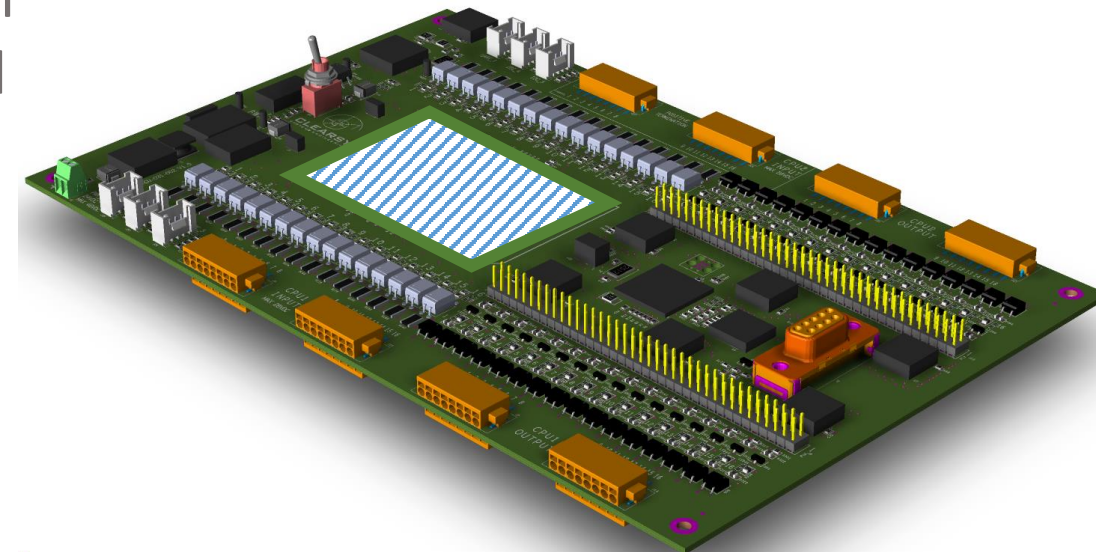
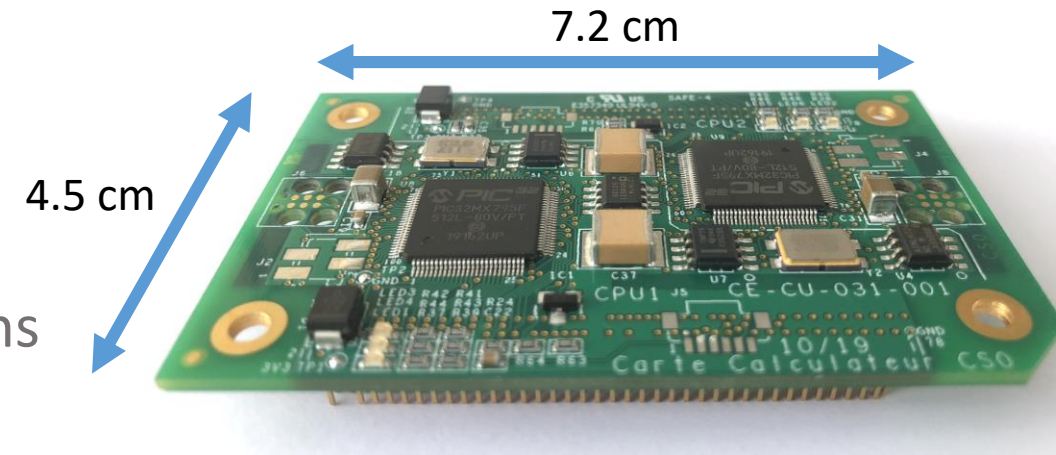
Schema_Edit



Model extraction from relay-based schematics

CLEARSY Safety Platform

- **Industry-strength commercial version (2020):**
 - Core (smartcard format) with 2x μC
 - Safety on hardware
 - Based on 2002 PIC32 microcontrollers
 - Offers up to 40 MIPS for lightweight applications
 - All PIC32 interfaces are available in order to address I/O, analog, communication bus, etc.
 - Safety on software
 - Based on 4004 software
 - **Correctness** is ensured by mathematical proof
 - Cross checks between software instances and between microcontrollers
 - To be plugged on a motherboard with
 - power supply,
 - (maintenance) processor,
 - network connection, and
 - I/O



References



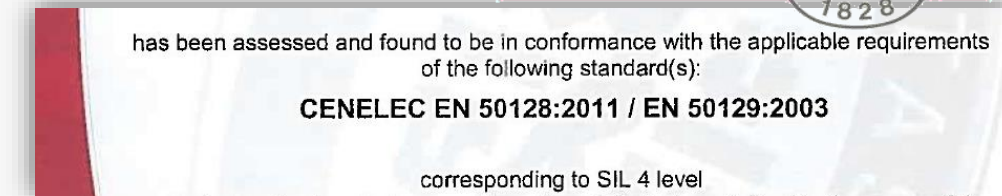
- ▶ 1st design of CLEARSY SIL4 processing architecture: for platform screen door operation, monorail Sao Paulo line 15



Generic product certificate, CERTIFER #8891/200-1
27th Feb 2017 **SIL4**



- ▶ Product fitted for Stockholm City Line platform screen door operation



System certificate BUREAU VERITAS #6393741
3rd March 2017 **SIL3**

- ▶ New design for CBTC input / output module (customer confidential)

Generic product certificate BUREAU VERITAS #7092509
23rd July 2019 **SIL4**

Also **AREMA** compliant (asserted by TÜV)

Conclusion & Perspectives

- ▶ Low-Cost safe execution platform for SIL4 application in research & development
- ▶ Hardware solution
 - ▶ Starter kit, board for prototyping
 - ▶ Layout/schematic ready **to integrate a design**
- ▶ Software solution
 - ▶ **IDE** to develop safety systems
 - ▶ tools to prove software compliance, to load and debug software
- ▶ Services
 - ▶ **Certification kit**, support for certification
 - ▶ Support for HW / SW design
 - ▶ design / industrialize specific safety products based on CSP

Conclusion & Perspectives

► Evolution

- **Improve proof** performance to support more complex algorithms
- PIC32 microcontrollers upgraded to **more powerful processors**
- Associated with (RTOS)(communication) processor

► Applications in

- Railways for lightweight applications (EN50129)
- Autonomous vehicles for low-level safety functions, safe infrastructure
- Industry for safety-related processes (IEC61508)
- **Legacy Systems** implemented with modern technologies



LOW COST HIGH INTEGRITY PLATFORM

THANK YOU FOR YOUR ATTENTION

<https://www.clearsy.com/en/our-tools/clearsy-safety-platform/>



Thierry Lecomte

David Deharbe

Patrick Péronne

Etienne Prun

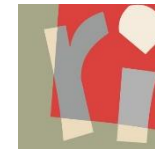
Denis Sabatier



Emmanuel Chailloux

Adilla Susungi

Steven Varoumas



Sylvain Conchon

Aims

- ▶ Development and deployment of safety-critical applications, **up to SIL4**
- ▶ An Integrated Development Environment and a hardware platform that **natively integrates safety principles**
- ▶ For Developing cyclic applications running directly on the platform without any underlying OS
- ▶ Drastically Reduce the time and effort to certify (80%)
- ▶ Fit for education and prototyping



▲ 2oo2 microcontrollers

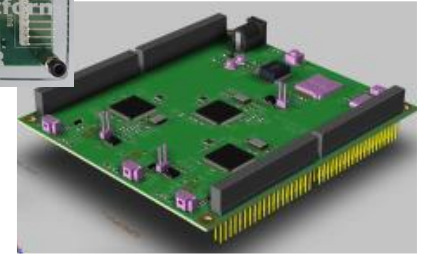
Function and Safety

Safety is built-in, out of reach of the developer who cannot alter it

- ▶ Code generation from a formal model (mathematical proof against its specification) and guarantee of no programming error.
 - ▶ But the implementable B model can also be obtained using other methods / languages
- ▶ Detection of divergent behavior (by this design)
- ▶ Based on a hardware/software dual processor architecture patented by CLEARSY

Fit for Education

- ▶ Available starter kits
- ▶ SK0: Starter Kit for school promotion campaign and industrial tests (3 digital Inputs, 2 Outputs in the board), ready for the market
- ▶ SK1: Starter Kit for complex system with the mother board including 8 Outputs, 20 Inputs



- ▶ Industrial software tools
- ▶ Based on Atelier B version 4.5 – Industrial Formal method
- ▶ Reduces deployment and certification costs
- ▶ Includes specific plugins to compile and load automatically to the platform.



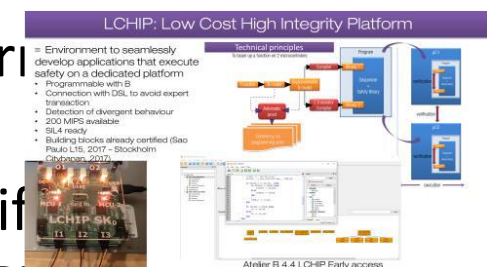
Main Features

Ready for industry

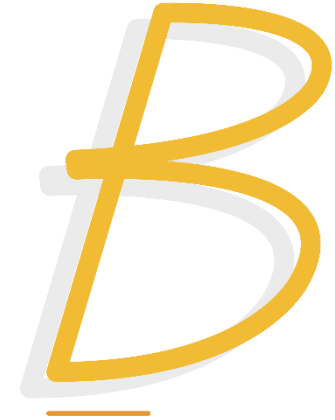
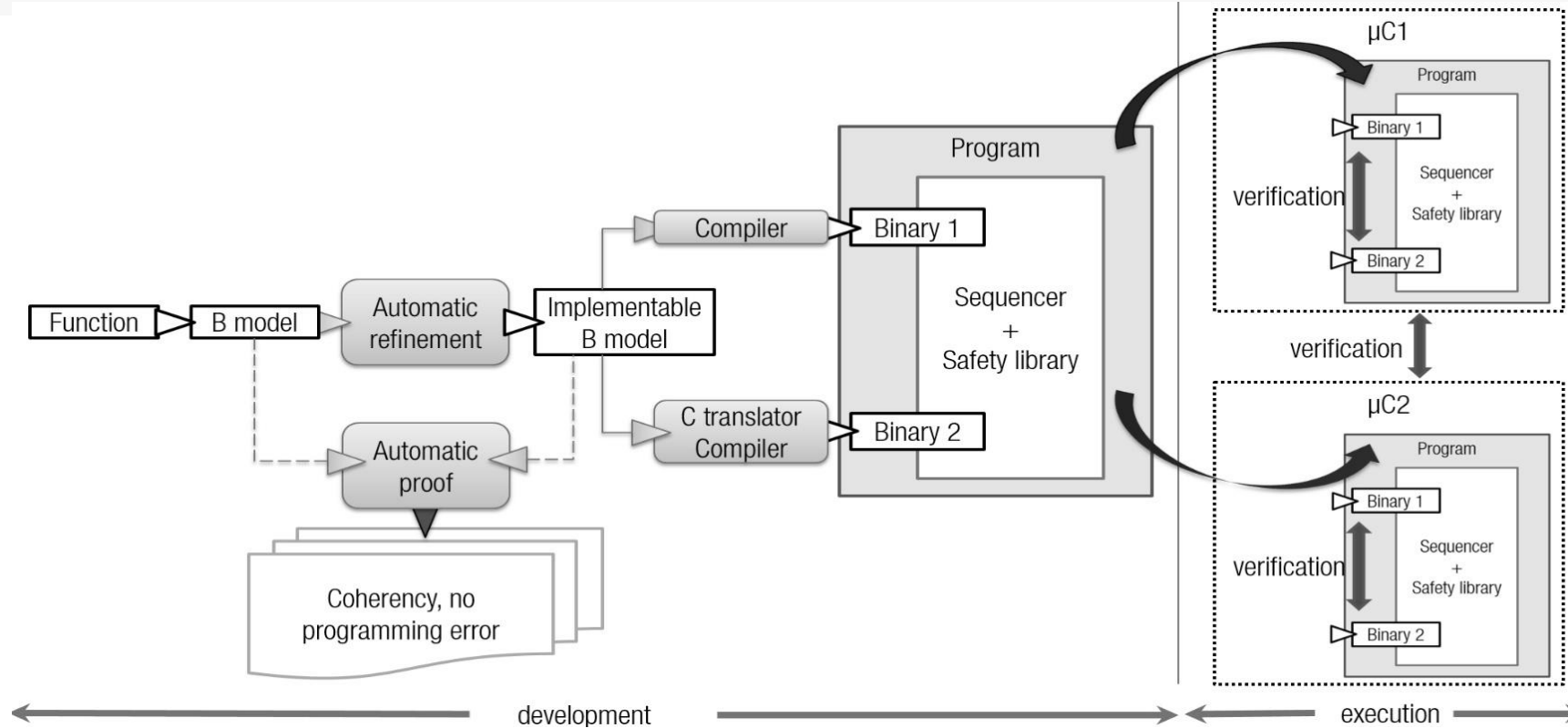
- ▶ Provided as a daughter board (8*5cm) to be included in in-house designs with certification kit
- ▶ The building blocks of this technology have already been **certified (SIL3 and SIL4)** in several railway projects worldwide
- ▶ **Safety principles are out of reach of the developer who cannot alter them**
- ▶ CLEARSY Safety Platform could be
 - ▶ adapted to another specific development process used by the customer
 - ▶ combined with other to improve availability or features

Dissemination and References

- ▶ Conferences
 - ▶ Conference FM 2018 (Formal Methods) – Oxford UK
 - ▶ Conference RSSR 20174 (Reliability, Safety and Security of Railway Systems: Modelling, Analysis, Verification and Certification) - Pistoia Italia
 - ▶ Ecole Doctorale ETMF 2017 (Escola de Informática Teórica e Métodos Formais) - Rio de Janeiro, Brazil
 - ▶ Conference SBMF 2017 (Brazilian Symposium on Formal Methods) – Recife, Brazil
 - ▶ Conference GRTMS (Global Conference on Signalling : the Evolution of ETCS) - Milan, Italy
-
- ▶ Training
 - ▶ Newcastle (UK - NewRail Centre for Railways Research) 02/2018
 - ▶ Montreal and Sherbrooke (Canada) - 04/2018
 - ▶ Niteroi / Pirnamirim / Natal (Brazil) - 05/2018



Double Compilation Chain



- ▶ Code generation from a formal model (mathematical proof against its specification) and guarantee of no programming error.
 - ▶ But the implementable B model can also be obtained using other methods / languages
- ▶ Detection of divergent behavior (by this design)
- ▶ Based on a hardware/software dual processor architecture patented by CLEARSY

Platform Composition

Safety on hardware

- ▶ Based on **2002 PIC32** microcontrollers
- ▶ Offers **up to 40 MIPS** for lightweight applications
- ▶ All interfaces from PIC32 are available in order to address I/O, analog, communication bus, ...



Safety on software

- ▶ Based on **4004 software**
- ▶ **Correctness** is ensured by mathematical proof
- ▶ Cross checks between software instances and between microcontrollers