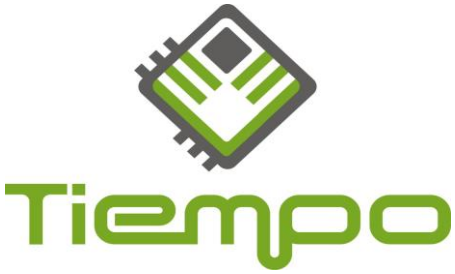


Model-checking Synthesizable SystemVerilog Descriptions of Asynchronous Circuits

Aymane Bouzafour[1], Marc Renaudin[1],
Hubert Garavel[2], Radu Mateescu[2], Wendelin Serwe[2]

[1]Tiempo Secure – SAS , Montbonnot-Saint-Martin, France.
[2]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG F-38000 Grenoble France

ASYNC2018

May 15th, 2018

Outline

- Context and Objectives
- Formal modelling and verification Flow
- “Memory Protection Unit” case study
- Conclusion and prospects

Context and Objectives

Context and Objectives

Context

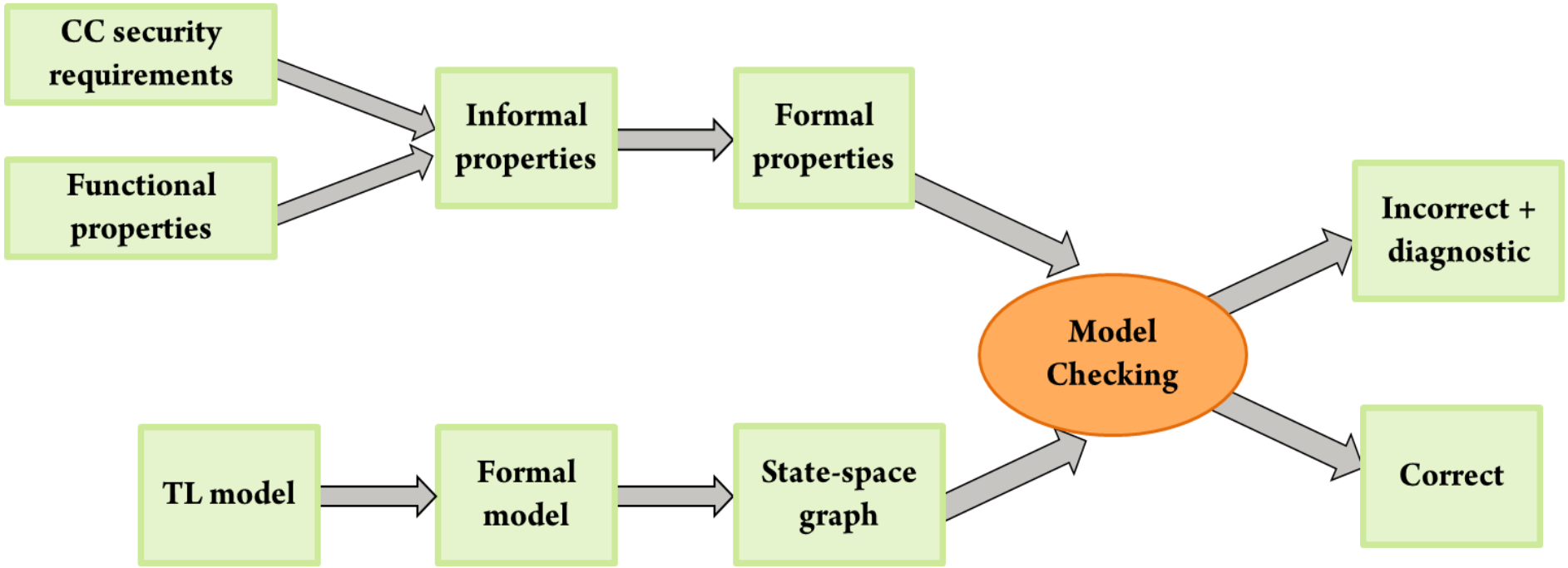
- **Tiempo's objectives for this work**
  - Enhancing the quality and security of Tiempo's products
  - Obtaining top-level certification for its products (Common Criteria EAL6/EAL7)

  ⇒ **Formal verification methods**

- **Election of CADP toolbox (INRIA/Convecs)**
  - Well suited for modelling asynchronous concurrent systems (process calculi based languages)
  - Long-standing collaboration and geographical proximity (CHP to Lotos translator…, DES…)

- **CADP toolbox**
  - A wide array of software tools for model checking and equivalence checking (more than 50 tools)
  - A long-run effort: Development of CADP started in 1986
  - Theoretical background: "concurrency theory"
  - Modelling languages: LOTOS (ISO 8807:1989), LNT derived from E-LOTOS (ISO 15437:2001), MCL
  - Numerous industrial utilizations for hardware verification (STMicroelectronics, Bull …)
  - Website: http://cadp.inria.fr

- Overview of the model checking endeavor

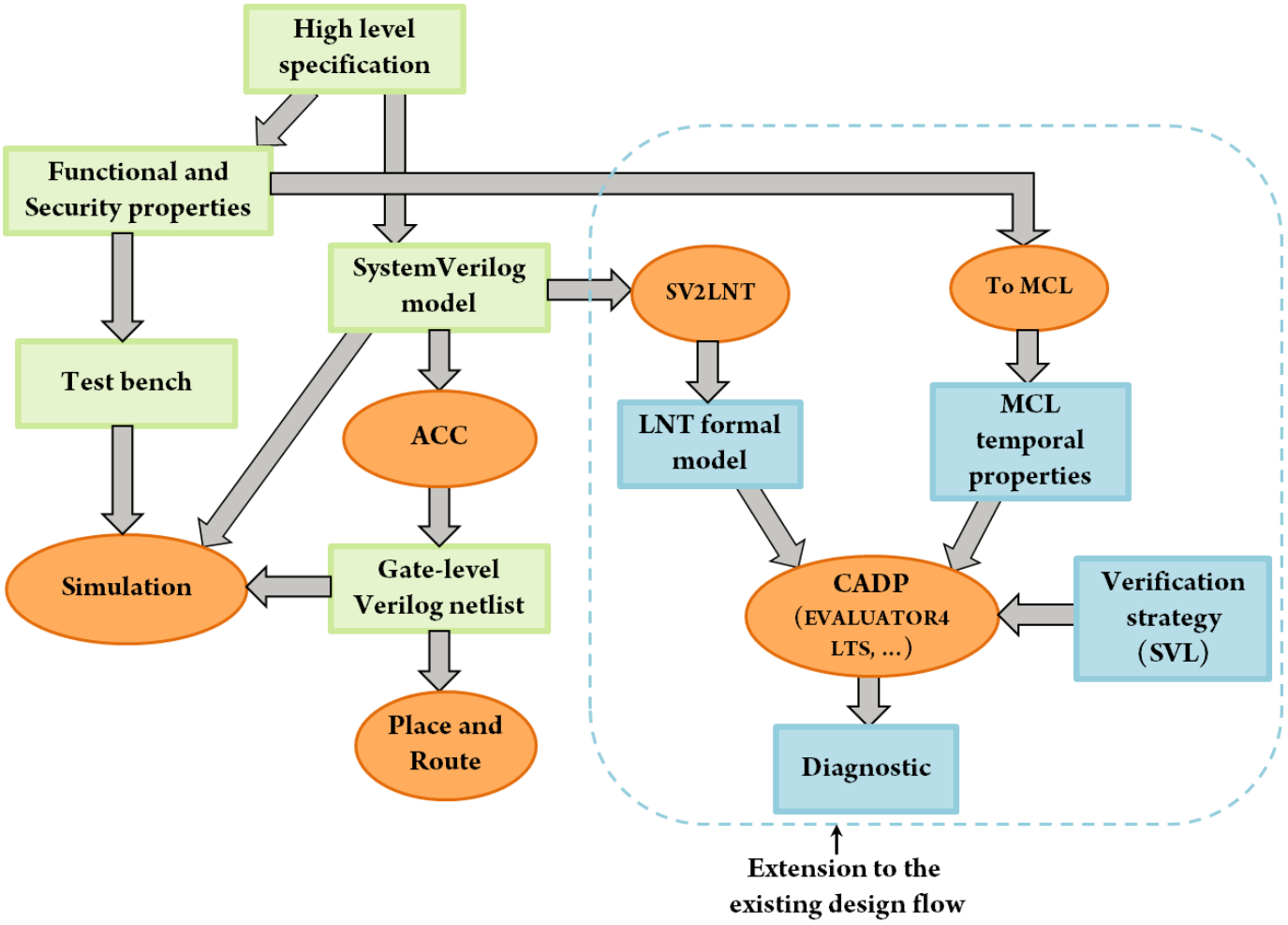


- **Results**
  - Formal verification of qualitative properties on the TLM of the asynchronous design
    - Functional properties
    - Security properties (Relating to the security requirements of third-party certifications)
  - Smooth integration of the specified formal modelling and verification flow in the existing Tiempo asynchronous design flow

Formal modelling and verification flow

Formal modelling and verification flow

- Supplemented Tiempo asynchronous design flow



- **Asynchronous SystemVerilog (ASV) [Renaudin et al.2012]**
  - Based on the standard HDL: SystemVerilog [IEEE 1800-2012]
  - Un-timed Transaction Level Modeling (TLM)
  - Concurrent processes communicating through channels
    - Channels => SV interfaces + handshake protocol
    - Compositions operators : Parallel (fork..join), Sequential (;)
  - Mixed Sync-Async design
  - Support by commercial CAD tools (simulation)

Formal modelling and verification flow
ASV to LNT translation

- **LNT**
  - Modern language for replacing LOTOS [ISO 8807:1989]
  - Similar features and coding style to ASV
- **Fully specified translation**
- **Done manually, but easy to automate**

```
-- main SV module
module address_decoder (
  ch_bit.in add_in,
  ch_data_t.in d_in,
  ch_data_t.out d_out0,
  ch_data_t.out d_out1
);
  always begin
    bit address;
    data_t data;
    fork
      add_in.BeginRead(address);
      d_in.BeginRead(data);
    join
    case (address)
      1'b0: d_out0.Write(data);
      1'b1: d_out1.Write(data);
    end case
    fork
      add_in.EndRead();
      d_in.EndRead();
    join
  end
end module
```

```
-- main LNT process
process main[
  add_in : ch_bit,
  d_in,
  d_out0,
  d_out1 : ch_data_t]
is
  loop var
    address : bit,
    data : data_t in
      par
        add_in(?address)
      || d_in(?data)
      end par;
      case address in
        0 -> d_out0(data); d_out0
      | 1 -> d_out1(data); d_out1
      end case;
      par
        add_in
      || d_in
      end par
  end var end loop
end process
```

Formal modelling and verification flow

Formal specification of properties

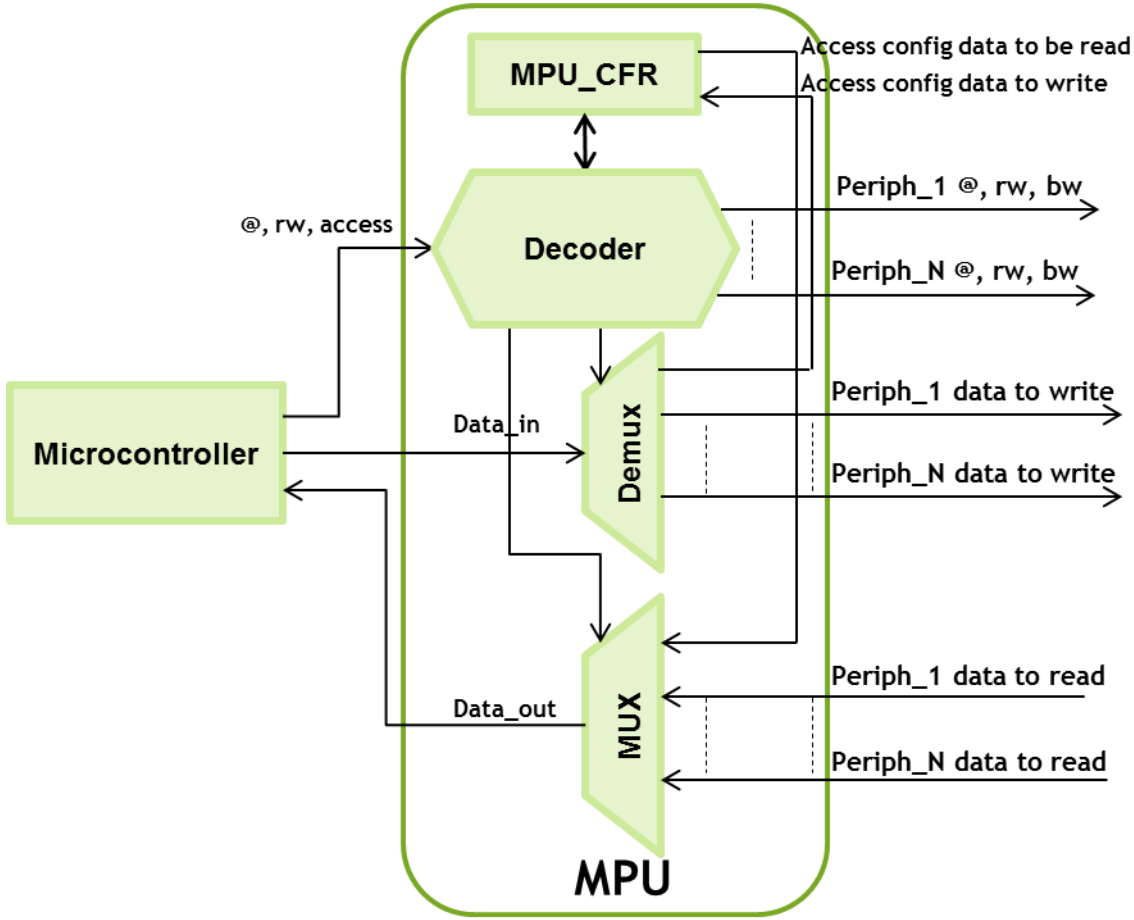
- **Temporal logic language: MCL**
  - Extended modal μ-calculus with (regular expressions, data-handling mechanisms…)
  - Expressing the qualitative properties as a combination of safety and liveness properties
- **Choice of MCL (MCL vs SVA, PSL)**
  - MCL is compatible with LNT and supported by CADP toolset
  - SVA and PSL are linear-time oriented and more suited to RTL properties

“Memory Protection Unit” case study

“Memory Protection Unit” case study

MPU Architecture

- A crucial block for security
  - Access-control
  - Security certifications
- Sufficiently complex (state-space wise)



"Memory Protection Unit" case study

Complexity of the MPU

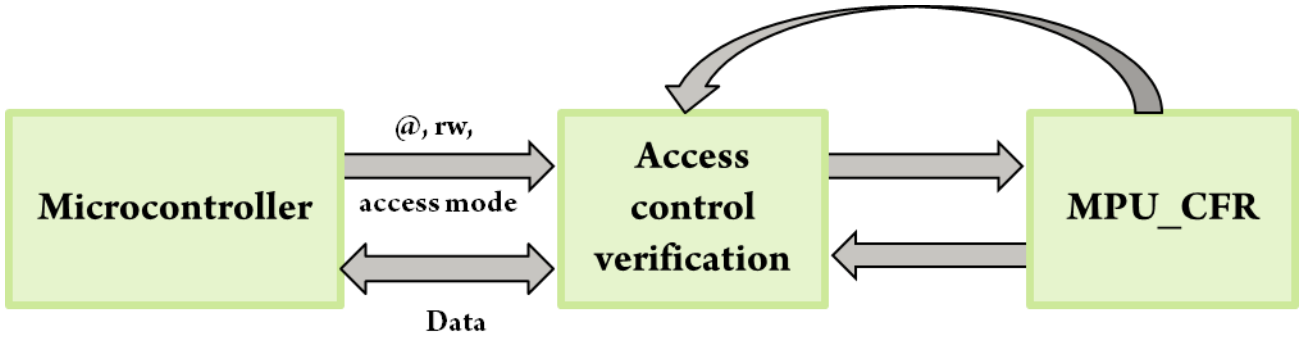
- **Size of the code**
  - 4400 lines of SystemVerilog
  - 8950 lines of LNT

  ⇒ **Modelling most ASV constructs and its typical design patterns**

- **High degree of internal concurrency**
  - 146 "main" concurrent processes (themselves concurrent)
  - 250 internal channels
  - 660 tokens in the underlying Petri net

  ⇒ **Exposing state-space related limitations**

- **Formal expression of the verified properties**
  - Expressing the verified functions as a combination of safety and liveness properties
  - Example: Access control CC requirement [CCPART2V3.1R5]
    - The SFR shall enforce the Memory Access Control Policy to restrict the ability to change initial values, modify or delete the security access rights of control information to running at privilege mode.



- **Formal expression of the verified properties**
  - Safety property
    - Unprivileged subjects cannot access/modify configuration registers (MPU_CFR)

```
-- MCL expression
[ "ACCESS_MODE_IN !0" . not ({ACCESS_MODE_IN ...})* . {CFG_REG1_WRITE ... } ] false;
```

  - Liveness property
    - A privileged subject requesting access to a configuration register must be granted access to the targeted register

```
-- MCL expression
[ PAR_3 ("ACCESS_MODE_IN !1", “ADDRESS_IN !BIT3_T(0, 1, 1)”, "WEN_IN !1") ]
INEVITABLE ( {CFG_REG1_WRITE ...} );
```

“Memory Protection Unit” case study

State-space reduction

- **State-space explosion**
  - Generation of the state-space
  - Exploring the state-space to verify properties
- **State space-reduction techniques**
  - Abstraction of data-types and variables
  - Reduction modulo graph equivalences (branching bisimulation)
  - Compositional state-space generation [Garavel et al. 2015]
    - "Divide-and-conquer" strategy
    - Projections and interfaces

“Memory Protection Unit” case study
Results

- **Verified properties**
  - Memory access control objective [CCPART2V3.1R5]
  - 184 qualitative properties verified
    - Functional properties (deadlock/livelock freedom, stimulus-response …)
    - Security properties (access-control policies)
- **Performance**

Acces type	Number of inter-mediate LTSs	Largest intermediate LTS		Final LTS			Time
		States	Transitions	States	Transitions	File size	
Co-processor	20	6.6 M	53 M	5.5 M	42 M	92 MB	13 min
MPU_CFR	20	27 M	355 M	27 M	355 M	692 MB	4h33
NVM	20	117 M	862 M	21 M	144 M	296 MB	3h34

- **Beyond the TESIC MPU**
  - The MPU verification is not a one-shot attempt
  - More designs are being verified by Tiempo:
    - Asynchronous Serial Link -- see model at [MCC'2018]
    - DES crypto-processor
  - Integration of CADP in Tiempo's design flow, usable by Tiempo hardware designers

Conclusions And Prospects

Conclusions and Prospects

- **Conclusions**
  - Appropriateness of the proposed formal verification approach
  - Validation on an industrial circuit by engineers relatively unfamiliar with formal verification

- **Prospective endeavors**
  - SystemVerilog to LNT translator
  - Extending the specified flow to lower abstraction-level descriptions
    - Formal verification of quantitative properties
    - Equivalence checking ASV vs Gate-Level

Thank you for your attention !