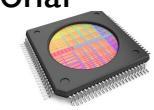
Runtime Assertion-Based Verification for Hardware and Embedded Systems

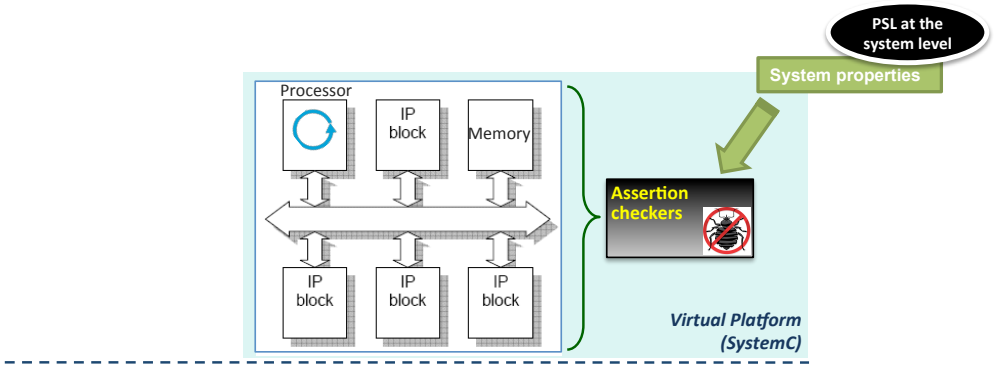
Laurence Pierre
Université Grenoble Alpes

Introduction

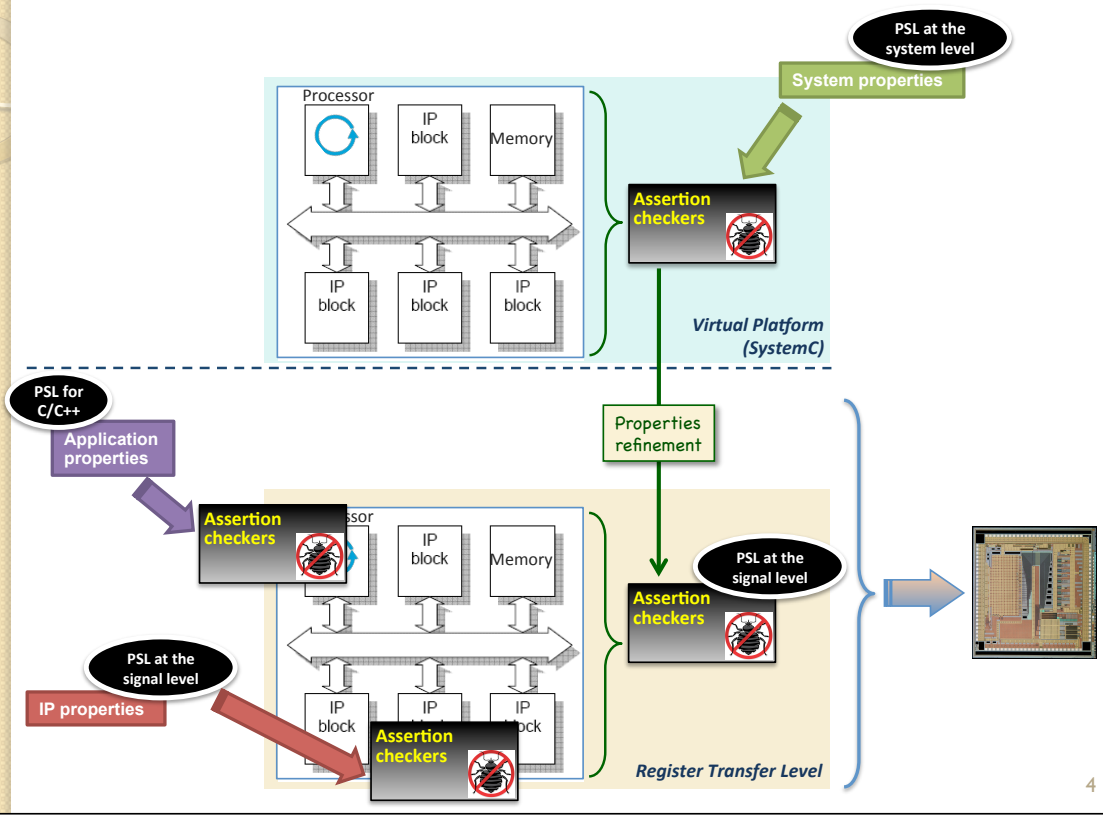
- Assertion-Based solutions at TIMA (context of embedded systems design)
  - Assertion-Based verification for PSL assertions
    - Property Specification Language, IEEE standard 1850, *https://standards.ieee.org/findstds/standard/1850-2010.html*
  - Runtime assertion-Based verification
    - For **hardware** components, at the transactional (TLM) and Register Transfer (RT) levels
    - For C embedded **software**



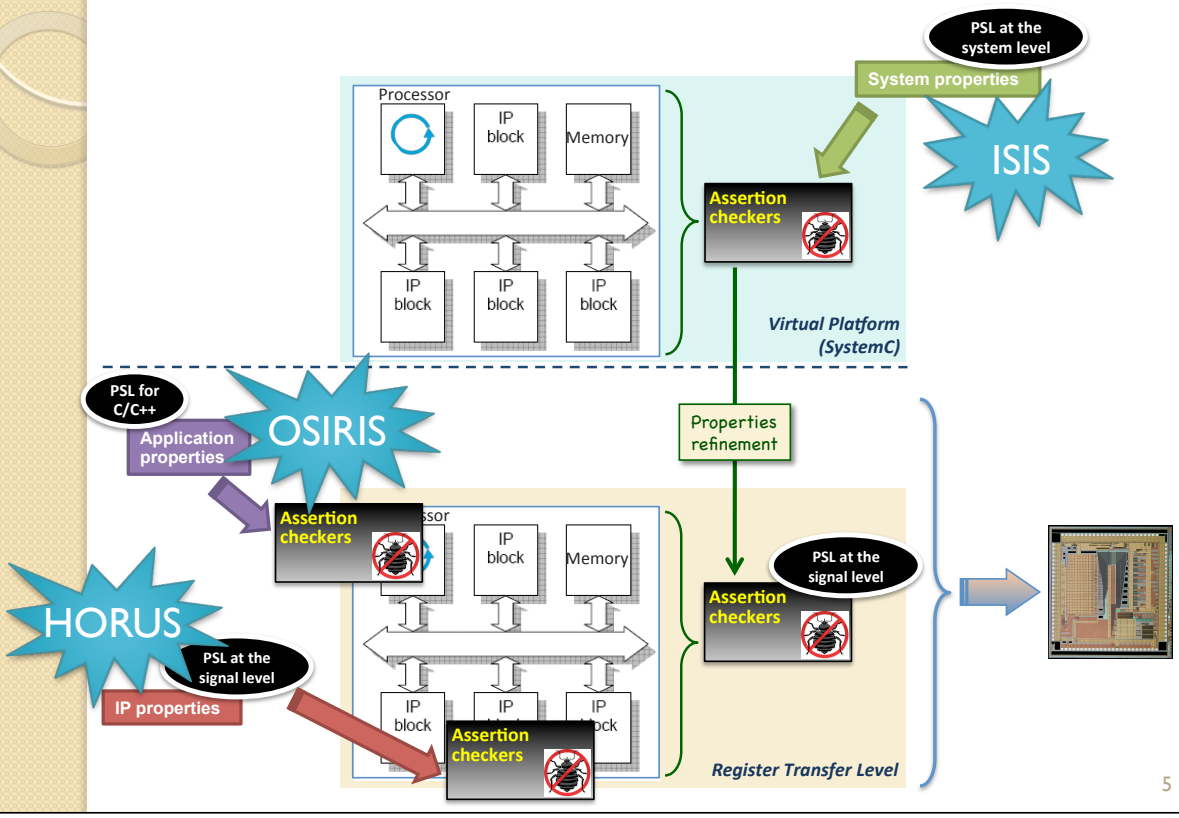
ABV solutions



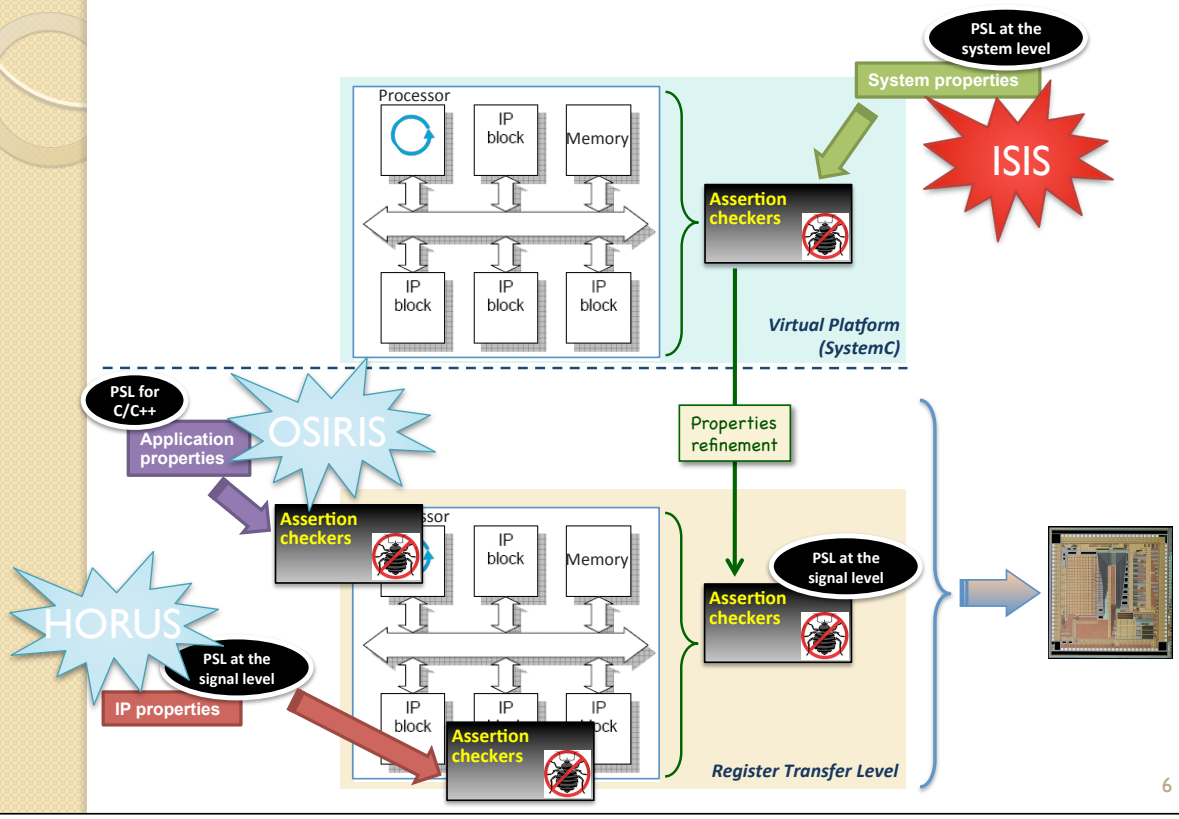
ABV solutions



ABV solutions



ABV solutions

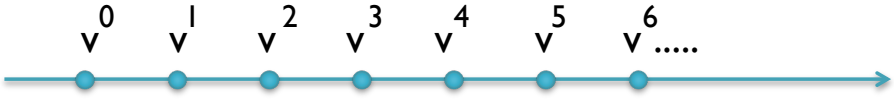


PSL semantics

- PSL (IEEE standard 1850)
  - The FL (Foundation Language) class of PSL formulas proposes temporal operators that are similar to the ones of **LTL**
- Semantics of FL properties
  - Defined with respect to **execution traces** i.e., words over the alphabet $2^{\mathcal{P}}$, where $\mathcal{P}$ is a set of atomic propositions
  - These execution traces are built by **sampling** the simulation on specific events

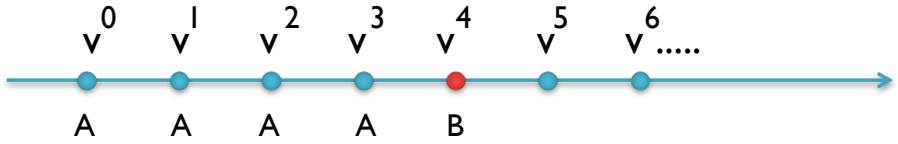
PSL semantics

- Semantics of some operators
  - $v \models b \Leftrightarrow |v| = 0$ or b holds on v^0
  - $v \models A \wedge B \Leftrightarrow v \models A$ and $v \models B$
  - $v \models$ **next!** $A \Leftrightarrow |v| > 1$ and $v^{1..} \models A$
  - $v \models A$ **until!** $B \Leftrightarrow$
    $\exists\, k < |v|$ s.t. $v^{k..} \models B$ and
    $\forall\, j < k,\ v^{j..} \not\models A$



PSL semantics

- Semantics of some operators
  - $v \models b \Leftrightarrow |v| = 0$ or b holds on v^0
  - $v \models A \wedge B \Leftrightarrow v \models A$ and $v \models B$
  - $v \models$ **next!** $A \Leftrightarrow |v| > 1$ and $v^{1..} \models A$
  - $v \models A$ **until!** $B \Leftrightarrow$
    $\exists k < |v|$ s.t. $v^{k..} \models B$ and
    $\forall j < k, v^{j..} \not\models A$



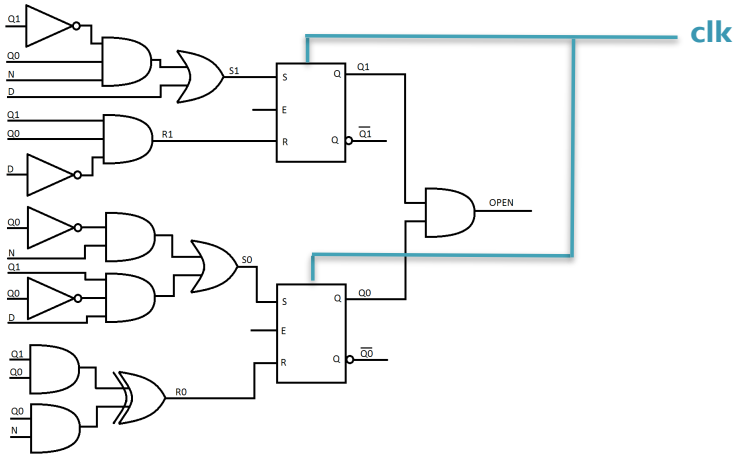
PSL semantics

- "Simple subset"
  - Dynamic (runtime) verification
  - Conforms to the notion of monotonic advancement of time
  - Rules
    - At most one operand of a logical or operator is a non-Boolean
    - The left-hand side operand of a logical implication (->) operator is a Boolean
    - The right-hand side operand of a non-overlapping until operator is a Boolean
    - ...

ABV at the RT level

HORUS

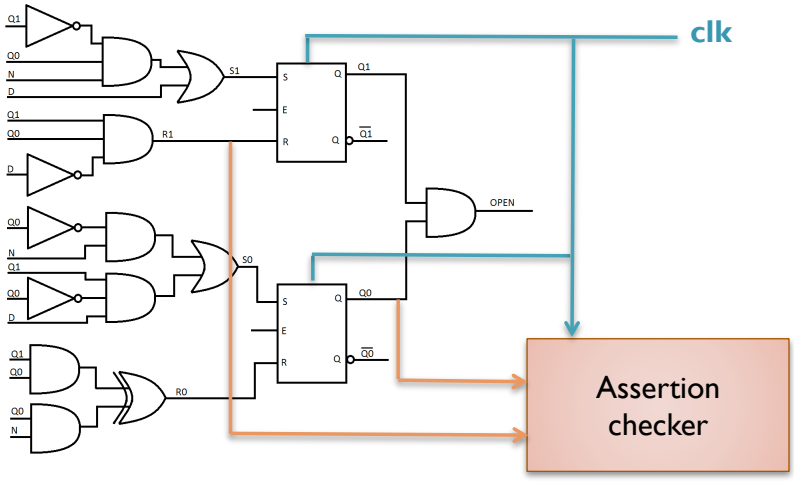
- RT and logic levels of abstraction
  - Example



ABV at the RT level

HORUS

- RT and logic levels of abstraction
  - Example

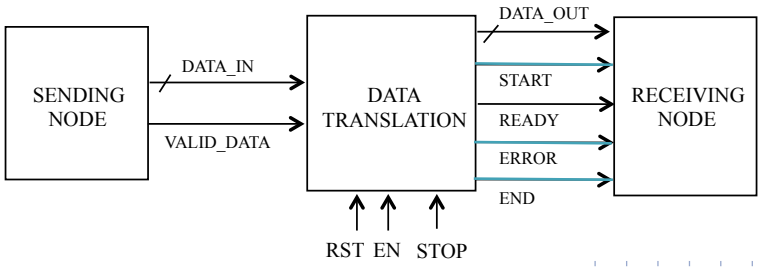


  - Sampling on clock edges

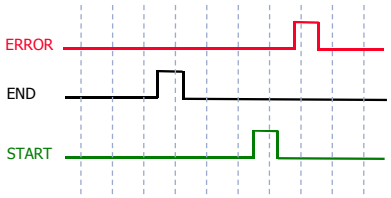
ABV at the RT level

HORUS

- HORUS: Runtime verification of RTL IP descriptions (e.g., VHDL)
  - Example



default clock = (posedge clk);
always(END ->
next (START before ERROR))

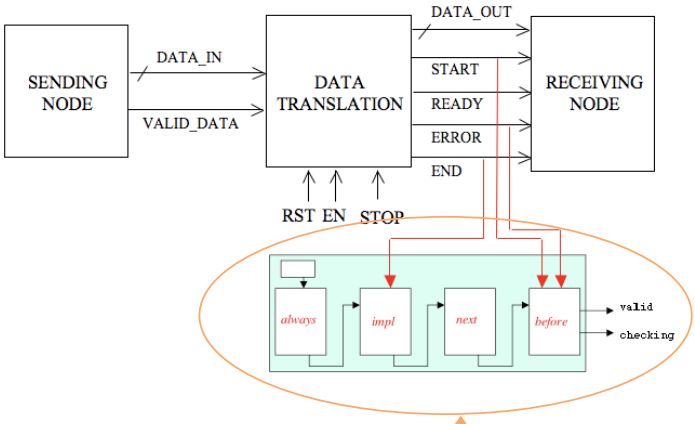


http://tima.univ-grenoble-alpes.fr/amfors/Horus/horus.html

ABV at the RT level

HORUS

- HORUS: Runtime verification of RTL IP descriptions (e.g., VHDL)
  - Example



always(END -> next (START before ERROR))

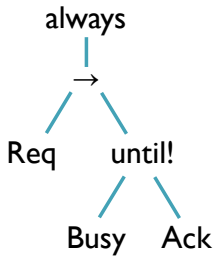
PSL assertion

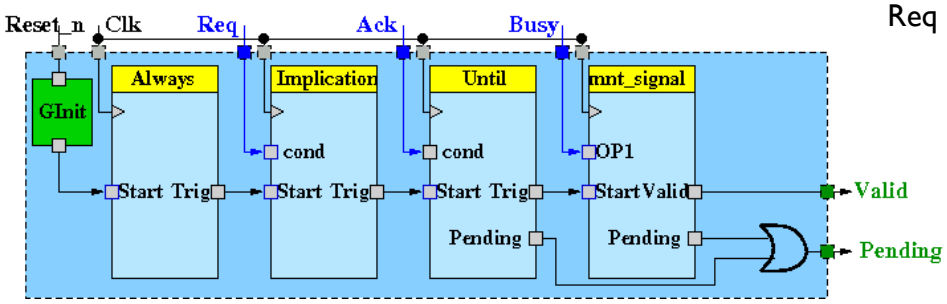
Checker

Construction of PSL checkers

HORUS

- At the RT level : compositional construction of VHDL components
  - Example
    - always (Req → (Busy until! Ack))

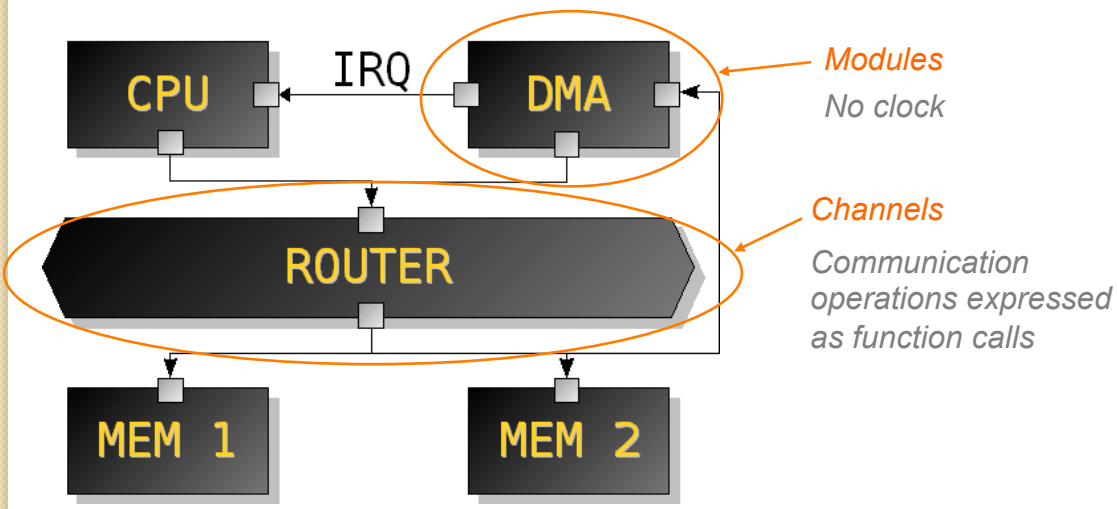




Y.ODDOS, K.MORIN-ALLORY, D.BORRIONE: "Assertion-Based Design with Horus". Proc. MEMOCODE'2008, June 2008

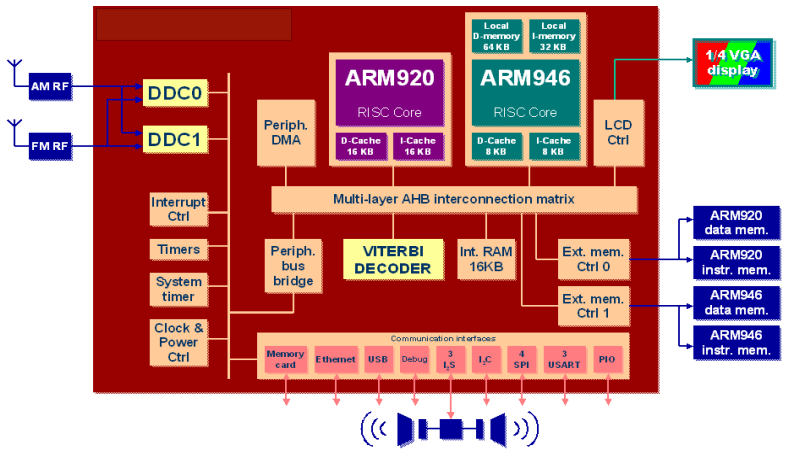
ABV at the transactional level

- Runtime verification of **virtual platforms** (SystemC models)



ABV at the transactional level

- Runtime verification of **virtual platforms** (SystemC models)
  - Example: Modem SoC for digital radio reception (Thales)



ABV at the transactional level

- The ISIS tool

ISIS

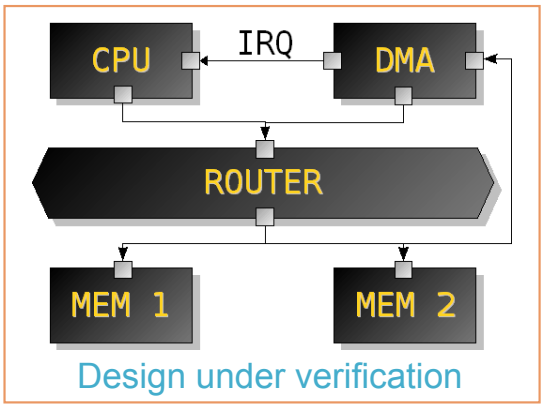
Instrumented simulation

PSL assertion

❶ *Automatic transformation*

SystemC assertion monitor (checker)

❷ *Automatic instrumentation*

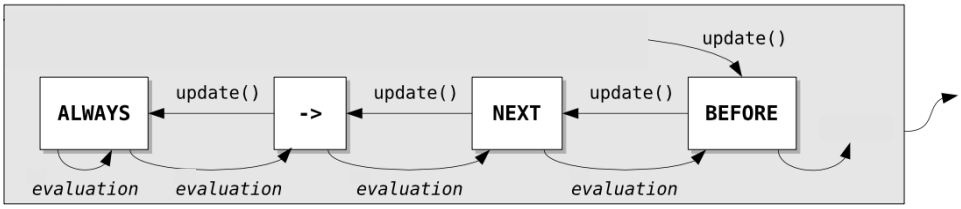


http://tima.univ-grenoble-alpes.fr/amfors/Isis/isis.html

Construction of PSL checkers

ISIS

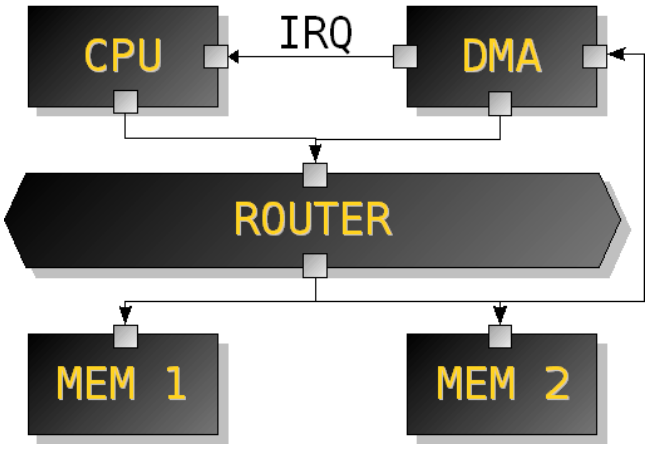
- Construction of C++ checkers
- Adaptation of the compositional construction method
  - Example
    - always ($exp_1 \rightarrow$ next (exp_2 before exp_3))



Instrumentation of TLM platforms

ISIS

- Sampling: Observation mechanism

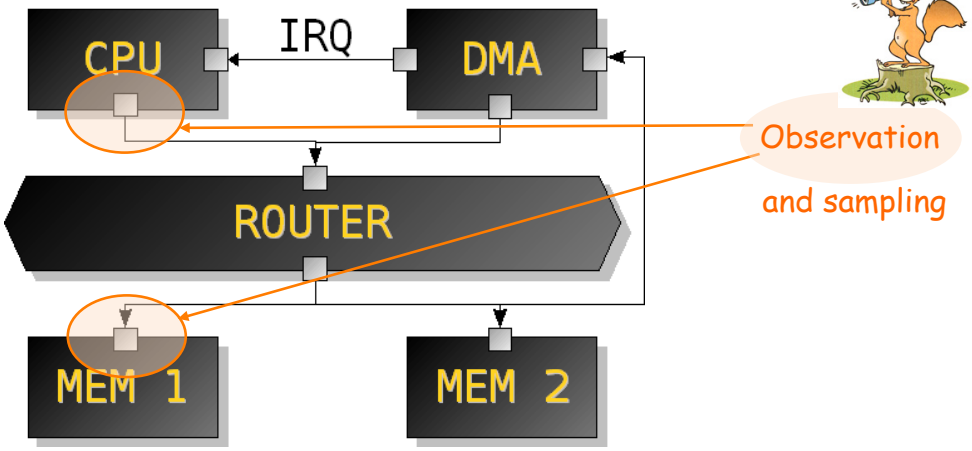


Any time a source address *is transferred* to the DMA, a *read access* eventually occurs and the right address is used

Instrumentation of TLM platforms

ISIS

- Sampling: Observation mechanism

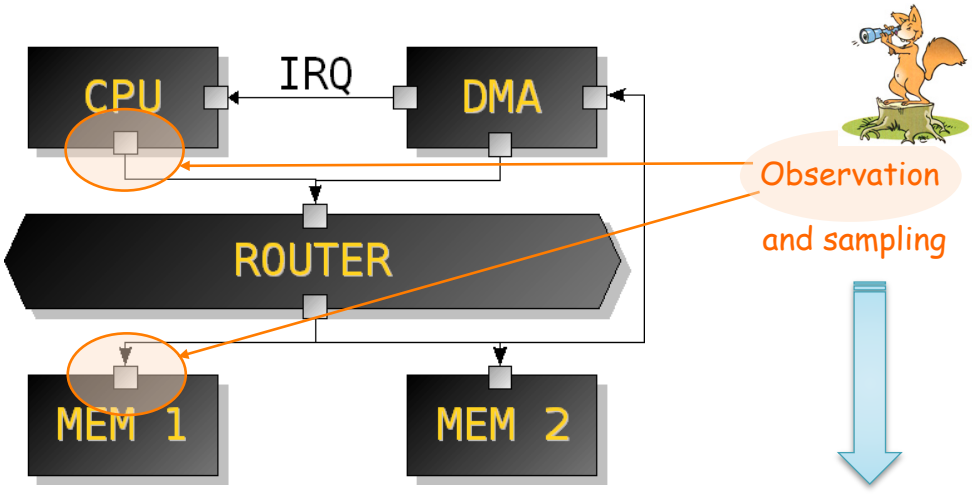


Any time a source address *is transferred* to the DMA, a *read access* eventually occurs and the right address is used

Instrumentation of TLM platforms

ISIS

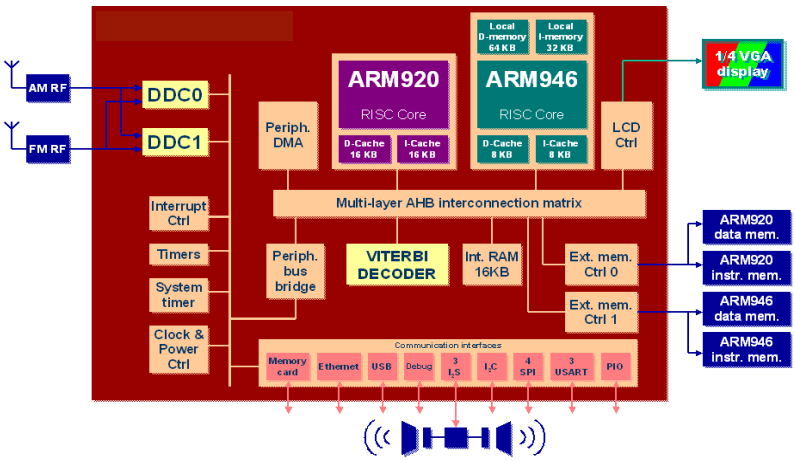
- Sampling: Observation mechanism



Any time a source address *is transferred* to the DMA, a *read access* eventually occurs and the right address is used

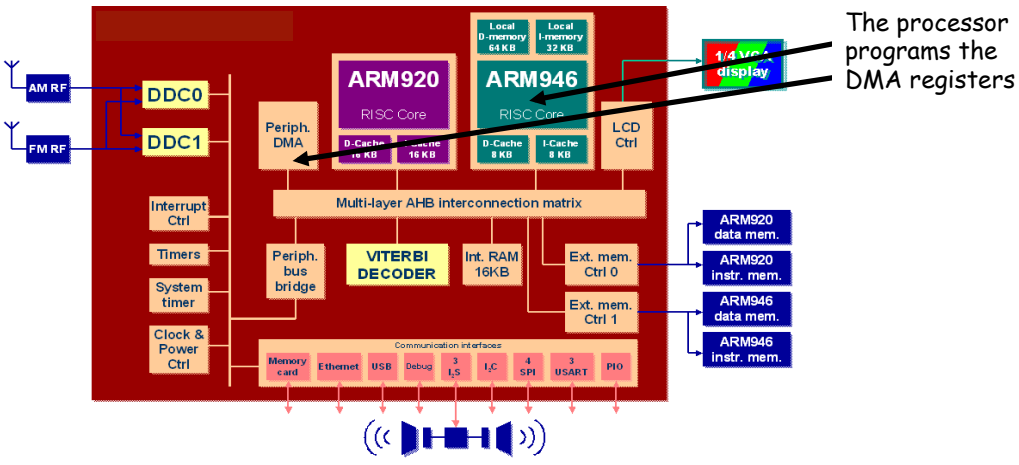
Instrumentation of TLM platforms

- Observation mechanism
  - Example:
    - Modem SoC for digital radio reception (Thales)



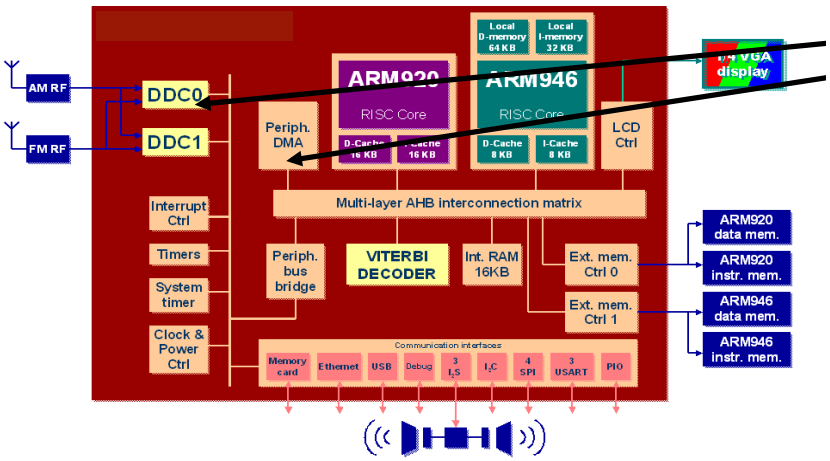
Instrumentation of TLM platforms

- Observation mechanism
  - Example:
    - Modem SoC for digital radio reception (Thales)



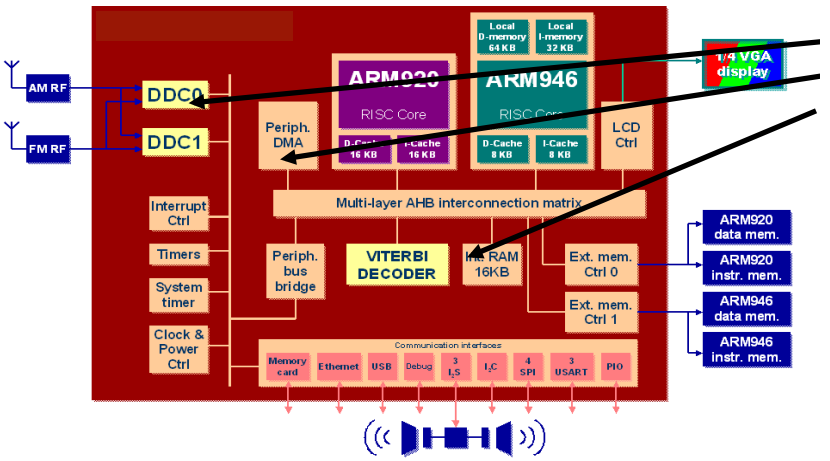
Instrumentation of TLM platforms

- Observation mechanism
  - Example:
    - Modem SoC for digital radio reception (Thales)



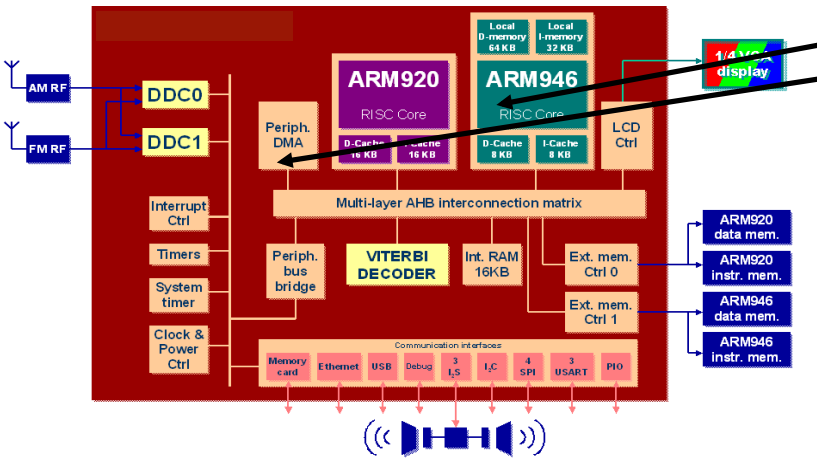
Instrumentation of TLM platforms

- Observation mechanism
  - Example:
    - Modem SoC for digital radio reception (Thales)



Instrumentation of TLM platforms

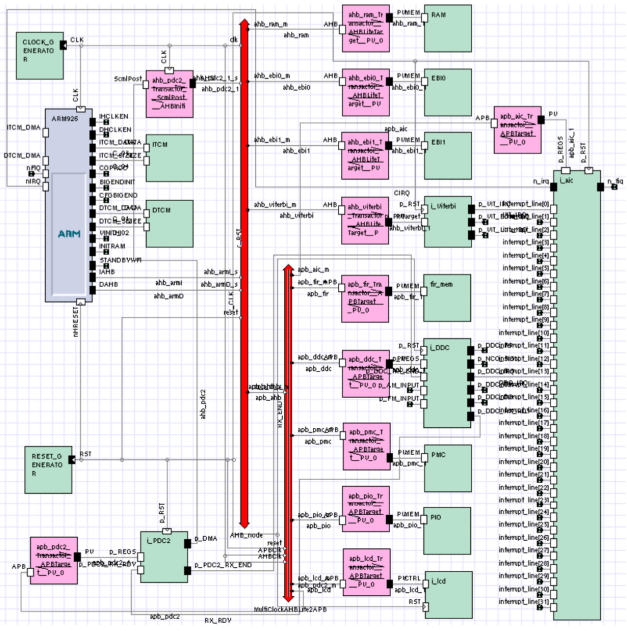
- Observation mechanism
  - Example:
    - Modem SoC for digital radio reception (Thales)



The processor processes the block of data and configures the DMA for the next transfer

Instrumentation of TLM platforms

- Observation mechanism
  - Example: Modem SoC



Instrumentation of TLM platforms

- Observation mechanism
  - Example: Modem SoC
    - « A reading of the interrupt controller register IVR clears the interrupt request sent to the processor »

→ *Whenever the register IVR of the interrupt controller is read, the interrupt request must be cleared before the next reading in the same register*

Reading at address 0x40 (IVR register) of the IRQ controller, through its target port

Signal CIRQ_pin is cleared

Instrumentation of TLM platforms

- Observation mechanism
  - Example: Modem SoC
    - « A reading of the interrupt controller register IVR clears the interrupt request sent to the processor »

Re-evalutated on every relevant events

```
always
  (targetaic.doPVtransport_CALL()
   && targetaic.doPVtransport.p1 == false
   && targetaic.doPVtransport.p2 == 0x40
   => next
        (CIRQ_pin
           before
              (targetaic.doPVtransport_CALL()
               && targetaic.doPVtransport.p1 == false
               && targetaic.doPVtransport.p2 == 0x40)));
```

Instrumentation of TLM platforms

- Observation mechanism
  - Example: Modem SoC
    - « There cannot be two consecutive DMA transfers at the same address in memory before a processor read »

→ *For every address in memory where the DMA writes, the processor must read at this address before the DMA writes again*

Reading through the ARM initiator port

Writing through the DMA initiator port

Instrumentation of TLM platforms

- Observation mechanism
  - Example: Modem SoC
    - « There cannot be two consecutive DMA transfers at the same address in memory before a processor read »

```
always
(Writing through the DMA initiator port
  => NEW(unsigned int add_write
                          = write address)
        (next ((Reading through the ARM initiator port
                && read address == add_write)
           before
           (Writing through the DMA initiator port &&
            write address == add_write))));
```

Instrumentation of TLM platforms

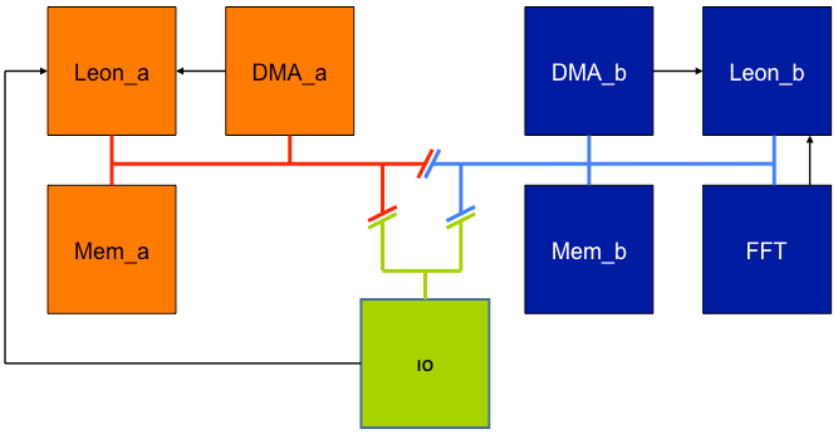
- Observation mechanism
  - Example: Modem SoC
    - « There cannot be two consecutive DMA transfers at the same address in memory before a processor read »

Re-evalutated on every relevant events

```
always
(dmainitport.post_write_CALL()
  => NEW(unsigned int add_write
                          = dmainitport.post_write.p4)
        (next ((arminitport.read_CALL()
               && arminitport.read.p1==add_write)
           before
           (dmainitport.post_write_CALL() &&
            dmainitport.post_write.p4 == add_write))));
```


Use case 1 - Image processing

- Image processing platform (Astrium)



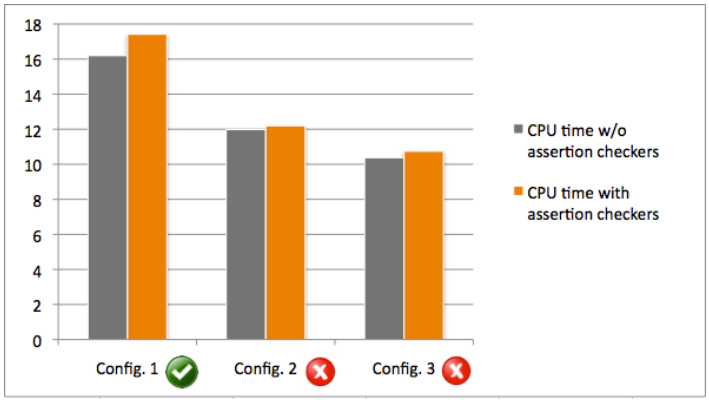
L.PIERRE: "Auxiliary Variables in Temporal Specifications: Semantic and Practical Analysis for System-Level Requirements". ACM Transactions on Design Automation of Electronic Systems (TODAES), Volume 21 (Issue 2), January 2016

Use case 1 - Image processing

- Some significant properties
  - DMA_a must not be configured before the end of the current transfer
  - The FFT module must not be configured before the end of the current computation
  - An input data packet must be read before the IO module generates a new interrupt
  - Each incoming data packet (read by DMA_a from the IO module) must have a corresponding output packet (written by DMA_b to the IO module), before 3 new incoming data packets are processed

Use case 1 - Image processing

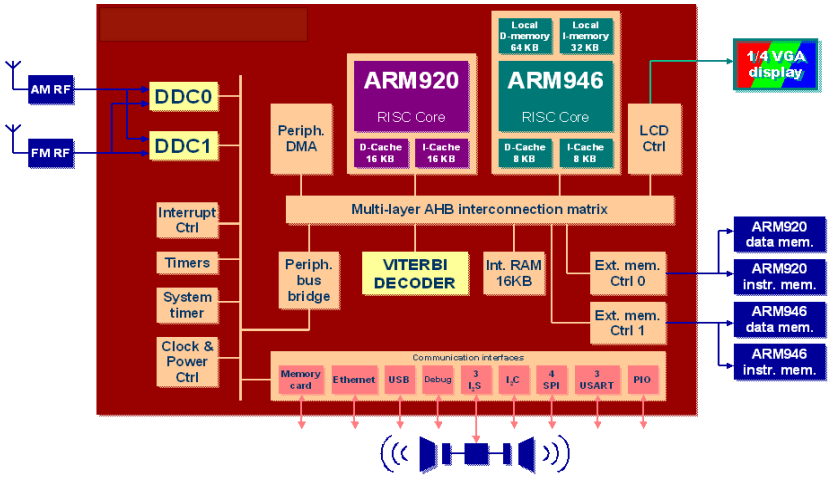
- Experimental results



- Config 1: bus frequency = 80 MHz, packet size = 4000 32-bits words
- Config 2: packet size = 5000 32-bits words
- Config 3: bus frequency lowered around 65 MHz, packet size = 5000 32-bits words

Use case 2 - Modem SoC

- Platform (Thales)



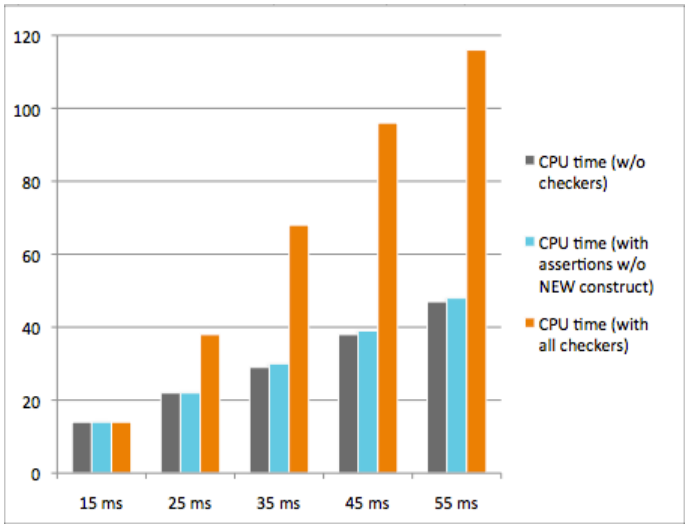
L.PIERRE, L.FERRO, Z.BEL HADJ AMOR, P.BOURGON, J.QUEVREMONT: "Integrating PSL Properties into SystemC Transactional Modeling - Application to the Verification of a Modem SoC". Proc. IEEE International Symposium on Industrial Embedded Systems (SIES'2012)

Use case 2 - Modem SoC

- Some significant properties
  - A reading of the interrupt controller register IVR (Interrupt Vector Register) clears the interrupt request sent to the processor
  - The DMA generates an interrupt between two transfer requests
  - There cannot be two consecutive DMA transfers at the same address in memory before a processor read
  - If the DDC has data to transfer, the DMA must initiate a transfer

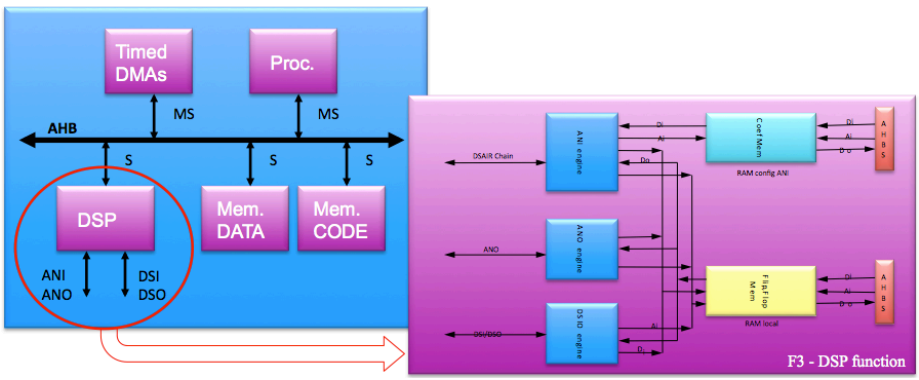
Use case 2 - Modem SoC

- Experimental results



Use case 3 - Flight control

- Avionics Flight Control Remote Module (Airbus)



L.PIERRE, M.CHABOT: "Assertion-Based Verification for SoC Models and Identification of Key Events". Proc. Euromicro Conference on Digital System Design (DSD'2017)

Use case 3 - Flight control

- Safety-related properties
  - Checksum computation must be performed every CHECKSUM_PERIOD ms (the corresponding result is stored in the register STATUS of the DSP)
  - The software must read the content of STATUS every CHECK_PERIOD ms
  - When a checksum error is detected (wrong value in STATUS), the DSP function must be deactivated (within LIMIT ms)

Conclusion

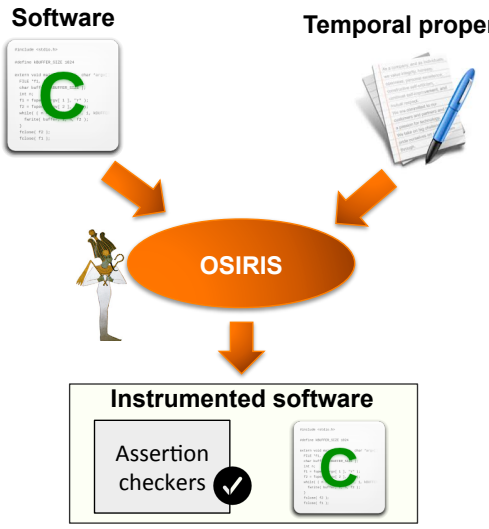
- ISIS : runtime verification of system-level temporal properties
- ... refined to be checked again at the RT level

  L.PIERRE, Z.BEL HADJ AMOR: "Automatic Refinement of Requirements for Verification throughout the SoC Design Flow". Proc. International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'2013)
- OSIRIS : runtime verification of temporal properties for embedded C programs – *On-going*

  M.CHABOT, K.MAZET, L.PIERRE: "Automatic and Configurable Instrumentation of C Programs with Temporal Assertion Checkers". Proc. ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE'2015)

Conclusion

- OSIRIS : runtime verification of temporal properties for embedded C programs



Conclusion

- OSIRIS : runtime verification of temporal properties for embedded C programs