Runtime Verification of Functional Requirements for SoC Models: Integration of PSL in SystemC TLM

Laurence PIERRE
TIMA Laboratory, Grenoble, France
Motivation – Verification

Is the communication channel reliable?

Does the DMA protocol work properly?
Motivation – Verification

Does the DMA protocol work properly?

Are data accessed as expected in the memory?

Is the communication channel reliable?

Does my software conform to given hardware-related constraints?
Introduction

• Main concern of this talk: validation of complex hardware/software SoC’s

• Answers to various questions:
  ▪ What is Assertion-Based Verification and how can it help?
  ▪ What does it mean at the System level and what are the capabilities of the ISIS tool?
  ▪ Applications?
    • to the verification of requirements for a SoC for space telemetry (Astrium) – FMICS’2011
    • to the verification of requirements for a SoC modem for digital radio reception (Thales Communications and Security) – SIES’2012

Assertion-Based Verification...

• Assertion: statement about the intended behaviour or a requirement of the design
  ▪ Temporal logics: CTL, LTL,...
  ▪ Specification languages: SVA (IEEE Std 1800), PSL (IEEE Std 1850)

• Assertion-Based Verification: does the design obey these temporal assertions?
  ▪ Static analysis (model checking)
  ▪ Dynamic verification (emulation, embedded checkers)
Assertion-Based Verification...

- **Assertion**: statement about the intended behaviour or a requirement of the design
  - Temporal logics: CTL, LTL, ...
  - Specification languages: SVA (IEEE Std 1800), PSL (IEEE Std 1850)
- **Assertion-Based Verification**: does the design obey these temporal assertions?
  - Static analysis (model checking)
  - Dynamic verification (emulation, embedded checkers)

ABV for IP Blocks

- **Assertion-Based Verification at the RT level**:
  - Fine-grained properties on the signals of the design
  - Example:

```vhdl
default clock= (posedge RxDataEn);
always( { not RxData ; RxData[*7] } )
|-> AbortFound before! StartOfFrame );
```
ABV at the System Level

- Modeling at the system level: SystemC TLM

- Observations:
  - The DMA generates an interrupt between two transfer requests

ABV at the System Level

- Assertions at the System level:

The DMA generates an *interrupt* between two *transfer requests*
Observation
Any time a source address is transferred to the DMA, a read access eventually occurs and the right address is used.
Every packet sent by a sender X to a receiver Y will eventually be delivered.
The ISIS Tool

- Verification of system-level requirements during simulation, using automatically generated checkers

SystemC design (TLM)

Simulation of the instrumented design

Requirements satisfaction?

Requirement (PSL assertion)

1. Automatic transformation

SystemC assertion monitor (checker)

2. Automatic instrumentation

Design under verification

Instrumented simulation
Example Simulation

- Instrumented simulation in Virtual Platform Analyzer (Synopsys)

Case Study 1

- SoC for space telemetry

SoCKET project - http://socket.imag.fr/
Case Study 1

- SoC for space telemetry – SystemC platform

Some typical requirements:

- Correct operation of the SW w.r.t. the HW
  - Example: *The processor does not start a new convolution processing before the completion of the previous one*

- Coding constraints imposed by quality rules or HW requirements are respected
  - Example: *Source and destination addresses must have been programmed before starting a convolution processing*

- Architectural properties respected by the SW
  - Example: *The memory does not respond with two “splits” consecutively to the same master*
Requirement 1 - Disambiguation

- The Leon processor does not start a new convolution processing before the completion of the previous one.

Writing in the read address register of the convolution unit.

Reading in the length register of the convolution unit a value that equals `image_size`.

```
always(leonPVinitiator_port.write_CALL() && leonPVinitiator_port.write.p1 == a_read_addr && leonPVinitiator_port.write.p5 != NO_CHECK => next ((leonPVinitiator_port.read_END() && leonPVinitiator_port.read.p1 == a_write_length && leonPVinitiator_port.read.p2 == image_size && leonPVinitiator_port.read.p5 != NO_CHECK) before (leonPVinitiator_port.write_CALL() && leonPVinitiator_port.write.p1 == a_read_addr && leonPVinitiator_port.write.p5 != NO_CHECK)));
```
Requirement 3 – Where to Observe?

• The memory does not respond with two “splits” consecutively to the same master – On the memory

```c
if (ahb_mem.transport_END()) {
    prev_master = master;
    master = ahb_mem.transport.p1.get_master_id();
    resp = ahb_mem.transport.p0;
    s = resp.get_ttp_status();
    status_split = (s.access_extension())->is_split();
}
else status_split = false;
```

```c
assert always((ahb_mem.transport_END() && status_split)
 => next (next_event
 (ahb_mem.transport_END() && (master == prev_master))
 (!status_split)));
```
CPU Time Overhead

- Requirements 1 and 2

- Requirement(s) 3
**Astrium’s Return of Experience**

- Good expressivity of PSL
- Moderate time overhead induced by monitoring
- The property checkers will provide a valuable help for non-regression testing
- Careful (natural language) expression of the requirements:
  - Disambiguate the properties, in particular the meaning of communication actions
  - Specify your preferred observation points

---

**Case Study 2**

- SoC modem for digital radio reception

---


---

SFINCS project - http://tima.imag.fr/vds/SFINCS
Case Study 2

- SoC modem for digital radio reception

The processor programs the DMA registers

The DDC reports data reception to the DMA

SFINCS project - http://tima.imag.fr/vds/SFINCS
Case Study 2

- SoC modem for digital radio reception

The DMA transfers data from the DDC to the memory, and sends an interrupt to the interrupt controller.

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

SFINCS project - http://tima.imag.fr/vds/SFINCS
Case Study 2

- SystemC model
  - ARM processor
  - AHB/APB bus
  - Interrupt controller
  - DMA
  - DDC
  + toy application for these experiments

Requirement 1 - Disambiguation

- « A reading of the interrupt controller register IVR clears the interrupt request sent to the processor »

- Signal CIRQ_pin

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

- When? Before the next reading in the same register
Requirement 1

- « A reading of the interrupt controller register IVR clears the interrupt request sent to the processor »

```java
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));
```

Requirement 2 - Disambiguation

- « 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

- Writing through the DMA initiator port
- Reading through the ARM initiator port
Requirement 2 - Disambiguation

- There cannot be two consecutive DMA transfers at the same address in memory before a processor read

```c
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)));
```

Violations detected with the toy application proposed for the experiments.

---

**CPU Time Overhead**

- Overhead can be significant with NEW properties (more than 270000 instances here)
- Negligible overhead for 3 "usual" properties
Thales’s Return of Experience

- SystemC: easy way to trace transactions and interactions, improves HW/SW design ability
- PSL: improves debug ability, easy expression of the HW or HW/SW expected behaviour
- In requirement 2:
  - A real-time constraint is not fulfilled -> Quick detection! 😊
  - Could have been “hidden away” from the experimenter by error-correction codes and redundancy in the signal
  - Joint information provided by gdb and by the assertion checkers can be particularly helpful to locate errors


Roadmap

- Formalisation of requirements
- FPGA prototyping and debug
- Synthesised properties + analysis infrastructure
- Assertions refinement
- Requirements (correctness or safety)
- Verification
- System on Chip (System level)
- RT Level
- M1
- M2
- M3
- Assertion checkers
- Embedded checkers
- Safety

FETCH’2013