Software for embedded systems

Verification Lab.
(Assertion-based verification)

Alessandro Danese

University of Verona, Verona, Italy
alessandro.danese@univr.it

January 15, 2019

Simple-platform case study

How to download the simple platform:

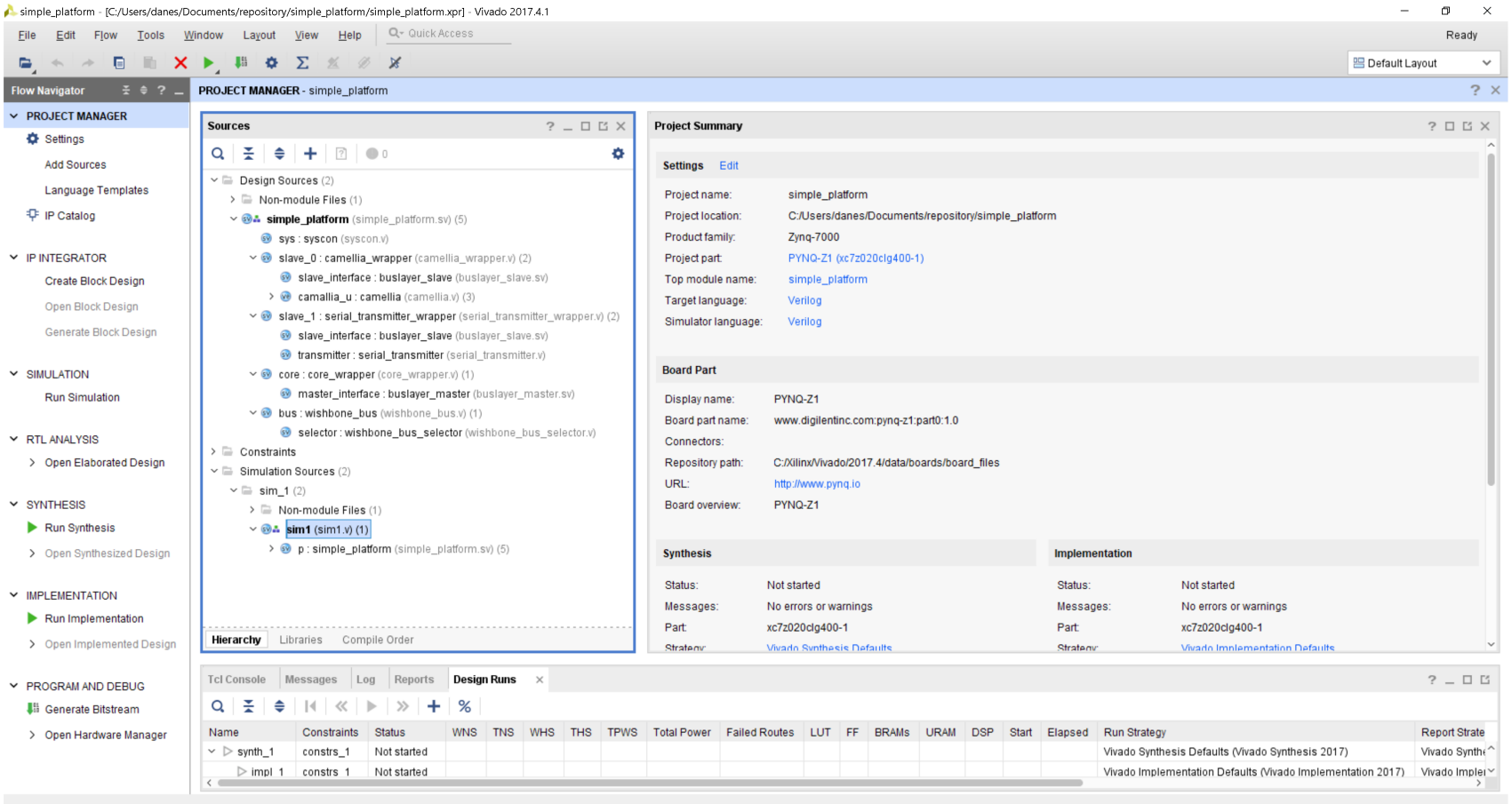
git clone https://AlessandroDanese@bitbucket.org/AlessandroDanese/sse-verifica.git

How to open the simple platform-project case study:

1. open Vivado
2. File -> Open project... -> path2/sse-verifica/simple-platform/simple_platform.xpr

Vivado

Once started, the following window should appear



EDA tools

How to download EDA tools:

1. scp esd-student@esd-srv01.scienze.univr.it:/tmp/esdlab.tar.gz (password: esd-student)
2. tar -xvf esdlab.tar.gz
3. cd esdlab
4. source start_eda.bash

Makefile menu

How to open the Makefile menu (terminal):

1. cd path2/sse-verifica/simple-platform/questa.simulation
2. make

Makefile menu

Once started, the following menu should appear on the terminal

```
================================================================================

USAGE: make RECEPIE|TARGET
Author: Alessandro Danese (alessandro.danese@univr.it)

--- RECIPES ------------------------------------------------------------------
simulation              => Performs: clean, compile_s, and run_s
mining_bl_master        => Performs: clean, and assertion mining for buslayer (master)
mining_bl_slave_0       => Performs: clean, and assertion mining for buslayer (slave_0)
mining_bl_slave_1       => Performs: clean, and assertion mining for buslayer (slave_1)
mining_camellia         => Performs: clean, and assertion mining for camellia
mining_transmitter      => Performs: clean, and assertion mining for transamitter
ABV                     => Performs: clean, and Assertion-based Verification
faultC_bl_master        => Performs: clean, and fault coverage for buslayer (master)
faultC_bl_slave         => Performs: clean, and fault coverage for buslayer (slave)
faultC_camellia         => Performs: clean, and fault coverage for camellia
faultC_transmitter      => Performs: clean, and fault coverage for transmitter

--- TARGETS ------------------------------------------------------------------
check_faults            => Performs: fault-coverage with faults.txt file
compile_s               => Compilings DUT
run_s                   => Runs simulation.

--- ADMINISTRATIVE TARGETS ---------------------------------------------------
help                    => Displays this message.
clean                   => Removes all intermediate and log files.

================================================================================
```

Simulating the simple platform

How to simulate the simple platform

1. make simulation

The command *make simulation* compiles the platform and the firmware source code (directory firmware).

Afterwards, it runs a simulation.
The sim.vcd file records the values of any register/wire/port of the platform. The transactor_log.txt file records any read_transaction and write_transaction performed by the firmware.

Assertion-based verification (ABV)

How to perform ABV with the simple platform

1. make ABV

The command *make ABV* compiles the source code of the platform, the firmware source code, and the verification unit in the files: buslayer_master.psl, buslayer_slave.psl, camellia.psl and transmitter.psl (sse_lesson1/vcs.simulation/psl).

Verification unit

A verification **vunit** is used to group PSL directives, and modeling code. A vunit is written in a side file that is bound to all the specified component's instances during the simulation.

```
vunit (component_name) {
...
}
```

Advantage:

- we do not change the source code of the component
- no glue logic to bind component's instances and properties
- properties and component's instances are connected automatically
- assertion failures are notified automatically

PSL directive

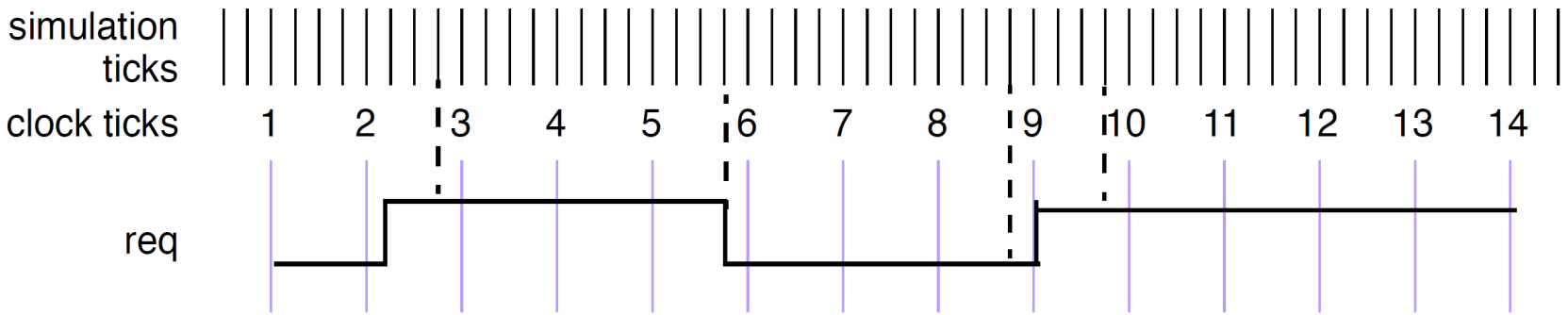
directive template

formula = [StrId:] directive (Property [@ clock]) [report Str];
directive = assert | cover | ...

Sampled value

The sampled value is the only valid value of a variable in a Property!

The sampled value is the value of a variable before a clock tick. A clock tick is an atomic moment in time that itself spans no duration of time.



req is high at clock ticks: 3, 4, 5, 10, 11, 12, 13, 14.
At 9th clock tick, req is still low!

Exercise - 1

Each file doc/*_spec.pdf contains a description of the functionality of the platform's components in natural language.

The current implementation of the simple platform does not meet all the listed specifications!

As a verification engineer, you are required to formalize the specifications of the platform's components in PSL, and find all the bugs!

Exercise - 2

As a verification engineer, you are required to formalize the behaviour of the component transmitter.

The transmitter component has not got a document describing its functionality. In this case, its behaviour can be inferred from its simulation trace, the mined assertions and from the source code.