# Software for embedded systems

## Verification Lab.
## (Fault coverage)

Alessandro Danese

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

January 15, 2019

# Simple-platform case study

How to download the simple platform:

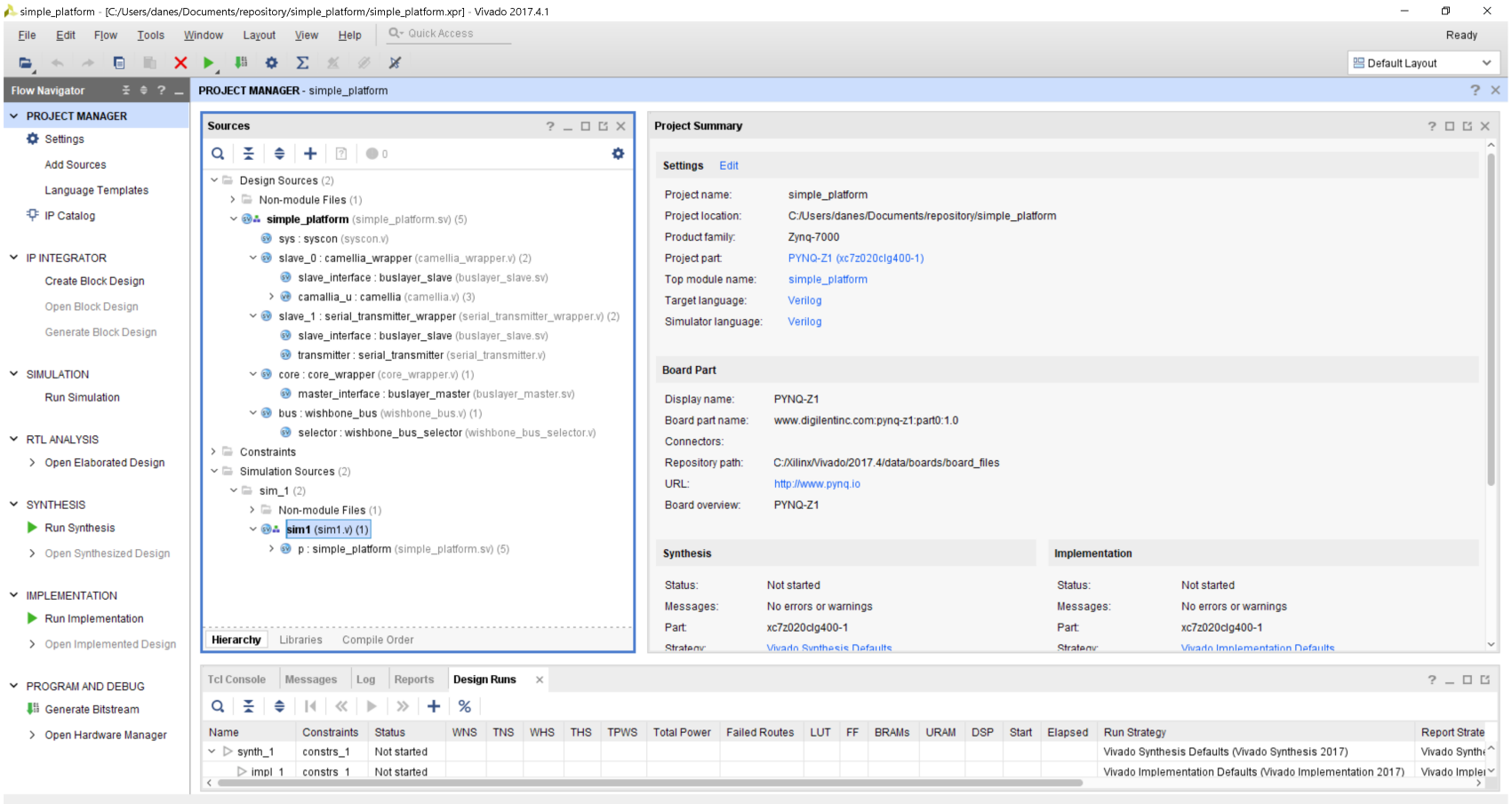
```
git clone https://AlessandroDanese@bitbucket.org/AlessandroDanese/sse-verifica.git
```

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.

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

# Force

The command force is a directive requiring the hardware simulator to force a value in a register/signal/port.

```
force <nid> <value>
       [<time> {, <value> <time>}* [-repeat <time>]]
       [-cancel <time>] [-freeze|-deposit] [-drive]

force <nid> -cancel <time>

where 'nid' is a nested identifier (hierarchical path name)
      'value' is the new value to be applied
      'time' is a [@]<number>[.<number>][<unit>]
         '@' identifies an absolute time
         'unit' is one of [ s | ms | us | ns | ps | fs ]
      -repeat <interval>
          repeat the waveform every relative time interval
      -cancel <time>
          release the forced value after the specified time
      -freeze|-deposit
          -freeze: default. Freeze the value to the forced value
          -deposit: value can be overwritten by a subsequent driver transaction
      -drive
          attach a new driver to the signal (VHDL only)
```

# Example

Forcing the bit input EN of camellia

- sim1.p.slave_0.camallia_u.EN 1'b0
- sim1.p.slave_0.camallia_u.EN 1'b1

Forcing the third bit of data input array of Transmitter

- sim1.p.slave_1.transmitter.data(2) 1'b0
- sim1.p.slave_1.transmitter.data(2) 1'b1

# Fault coverage

How to perform fault coverage with the simple platform.

## Example transmitter

1. cd sse_lesson1/vcs.simulation
2. make faultC_transmitter

The command *make faultC_transmitter* injects forces to simulate stuck-at faults. The generated file coverage.txt reports the assertions failed for each injected fault.

# Exercise - 1

Define a set of PSL properties for the transmitter component such that any fault location listed in questa.simulation/faults/transmitter_faults.txt is covered by at least a property.

# Exercise - 2

Define a set of fault locations for the components buslayer_master, and buslayer_slave.

For each component, define a set of PSL properties such that any fault is covered by at least a property.