## Software for embedded systems

Verification Lab. (Fault coverage)

#### Alessandro Danese

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

January 21, 2019

# Simple-platform case study

How to download the simple platform:

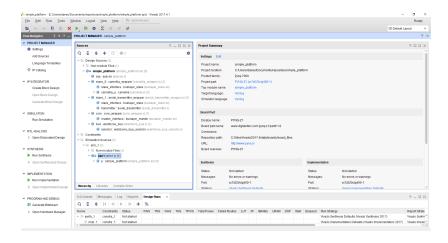
git clone https://Aless and ro Danese @bitbucket.org/Aless and ro Danese/sseverifica.git

How to open the simple platform-project case study:

- open Vivado
- 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:

- scp esd-student@esd-srv01.scienze.univr.it:/tmp/esdlab.tar.gz (password: esd-student)
- tar -xvf esdlab.tar.gz
- cd esdlab
- source start\_eda.bash

## Makefile menu

How to open the Makefile menu (terminal):

- od path2/sse-verifica/simple-platform/questa.simulation
- make

#### Makefile menu

## Once started, the following menu should appear on the terminal

```
USAGE: make RECEPIE TARGET
Author: Alessandro Danese (alessandro.danese@univr.it)
--- RECTPES ------
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
                      => Performs: clean, and Assertion-based Verification
ΔRV
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 -----
 \begin{array}{lll} \mbox{check\_faults} & \mbox{=> Performs: fault-coverage with faults.txt file} \\ \mbox{compile\_s} & \mbox{=> Compilings DUT} \\ \end{array} 
                      => Runs simulation.
run s
--- ADMINISTRATIVE TARGETS ------
heln
        => Displays this message.
      => 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

- od sse\_lesson1/questa.simulation
- 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 fault locations for the components transmitter.
- Oefine a set of assertions for the component transmitter.
- Run a fault coverage analysis to check if all injected faults are covered.
- Repeat steps 2 and 3 until all injected faults are covered by at least an assertion.