SystemC Co-simulation

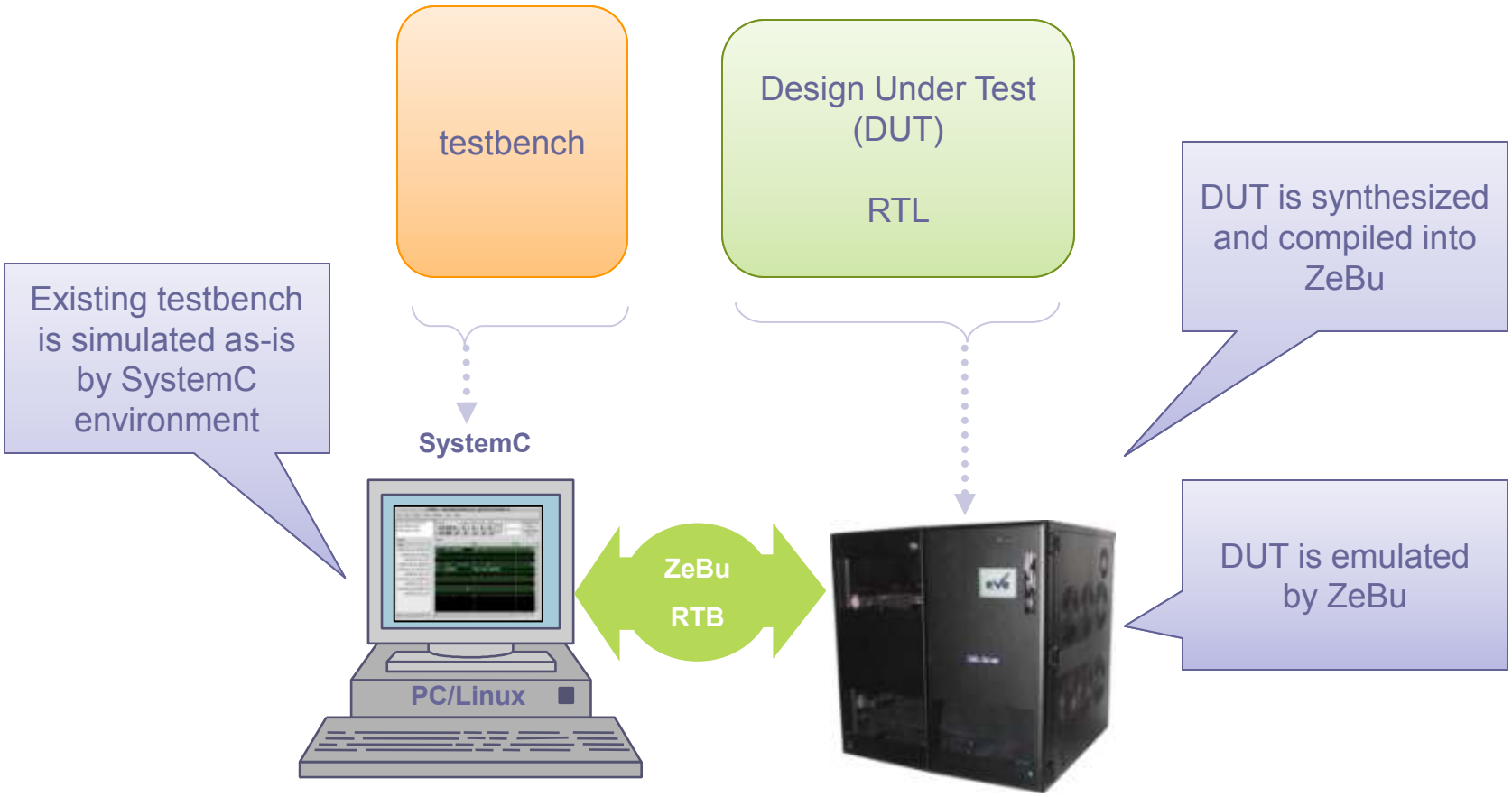
ZeBu-Server Training

*THE **FASTEST** VERIFICATION*

Agenda

- Overview
- Compilation
- Writing the testbench
- Debug

Overview



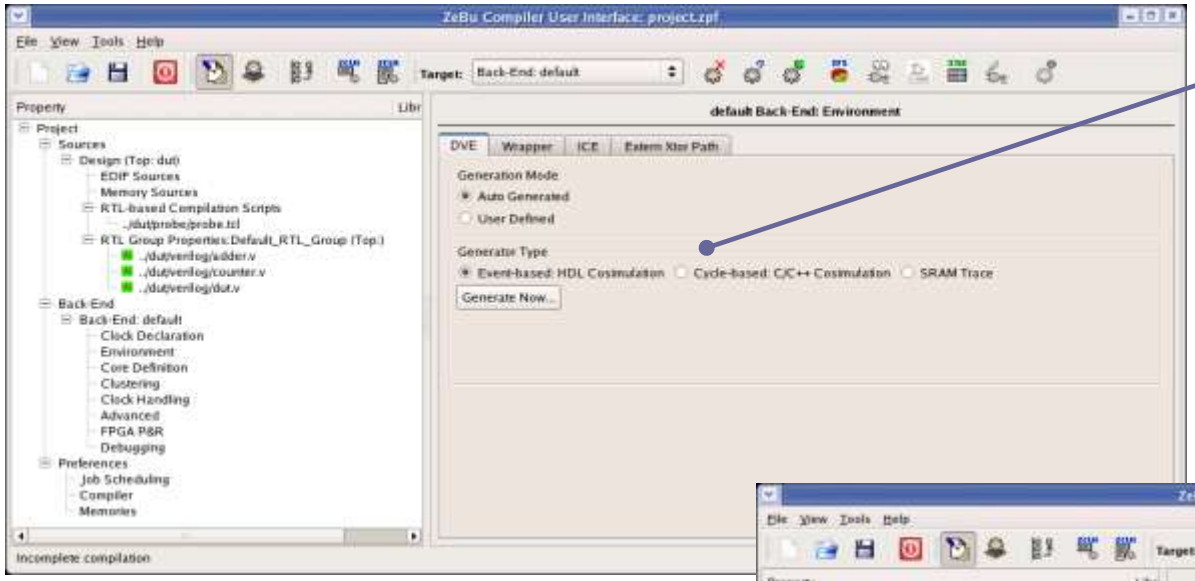
Overview

- The design is replaced by a SystemC wrapper which is generated during the compilation
  - The wrapper is used to connect the testbench to the ZeBu system
  - Replacing the design model by the generated wrapper and including the wrapper include file in the testbench is enough to get access to all interface and probe signals
- The co-simulation runs in event-based mode

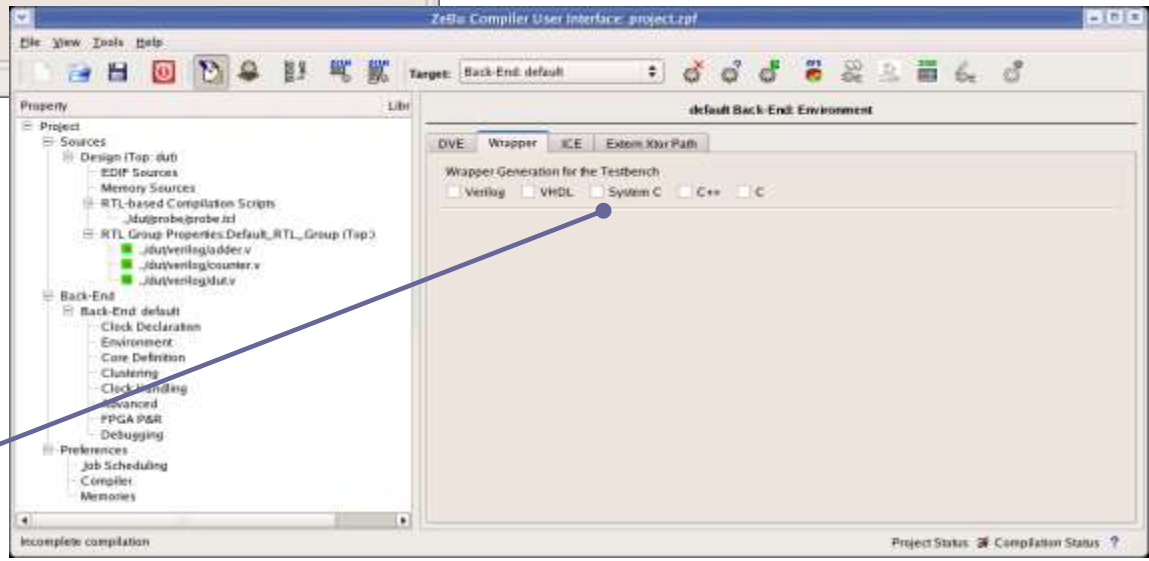
Agenda

- Overview
- Compilation
- Writing the testbench
- Debug

Compilation
zCui



Select operation mode:
Event-Based, HDL co-simulation



Select SystemC

Compilation
Generated files

- The compilation flow will generate SystemC wrapper files named `<top>.hh` and `<top>.cc`
- These files provide access to the HDL co-simulation driver
  - `<top>.hh` must be included in the testbench
  - `<top>.cc` must be compiled and linked with the testbench
- Including the generated header file in the testbench provides an access to interface signals, static and dynamic probes

Agenda

- Overview
- Compilation
- Writing the testbench
- Debug

Writing the testbench

- To connect the testbench (running in the SystemC environment) to the DUT (running in the ZeBu system) you need to:
  - Use the `libZebuSystemC.so` library
  - Replace the DUT original model by the wrapper generated during compilation
  - Include the `<top>.hh` file in the testbench
  - Define the `ZEBU_WORK` Linux environment variable
    `export ZEBU_WORK=../zcui.work/zebu.work`
- A `try/catch` instruction is highly advised
- To use 4-state values uncomment the following line in the `<top>.hh` file:
  - `//#define _LOGIC_VALUES_`
  - Will use 2-state values otherwise

Compiling the testbench

- Compilation of SystemC testbenches is done using GNU Compiler g++
- Use correct g++ version:
  - g++ 3.4, RedHat Enterprise Linux 4
- Correct include path must be given to compiler:

```
g++ -c zcui.work/zebu.work/top.cc -I$ZEBU_ROOT/include
   -Izcui.work/zebu.work -I$SYSTEMC/include
g++ -c testbench.cc -I$ZEBU_ROOT/include -Izcui.work/zebu.work
    -I$SYSTEMC/include
```

- Correct linking path must be given to linker:

```
g++ -o testbench top.o testbench.o -L$ZEBU_ROOT/lib
   -lZebuSystemC -L$SYSTEMC/<arch> -lsystemc
```

Agenda

- Overview
- Compilation
- Writing the testbench
- Debug

Debug

- When debugging a design in SystemC co-simulation mode you mainly use the SystemC environment debug features:
  - Waveforms
  - step-by-step in the testbench source code using a software debugger such as `gdb`
- In addition, you can use the following on the ZeBu side to monitor DUT internal signals:
  - Static probes
  - Dynamic probes
  - Simulated combinational signals
- You can also access ZeBu memories from the testbench

Debug
Static probes

- The wrapper file mimics the original design hierarchy, so signals probed using static probes are visible at their original location
  - E.g.: `top.u1.u2.signal`

Debug
Dynamic probes

- The wrapper file mimics the original design hierarchy, so signals probed using dynamic probes are visible at their original location
  - E.g.: `top.u1.u2.signal`
- Accessibility can be turned on and off to speed up emulation. When accessibility is turned off, dynamic probes display an 'X'
- Accessibility can be turned on and off as follows:

```
<top>.readbackOn();      // Switch on
<top>.readbackOff();     // Switch off
```

Debug
Simulated combinational signals

- Runtime behavior of simulated combinational signals is similar to dynamic probes and therefore the use model is the same

Debug
Accessing ZeBu memories

- **To access a ZeBu memory during SystemC co-simulation include the ZeBu C/C++ API in the testbench:**

```
#include <libZebu.hh>
```

- **Then use the C/C++ memory access API:**

```
<top>.<memory>->loadFrom("memFile");        // load
<top>.<memory>->storeTo("memFile");         // dump
```

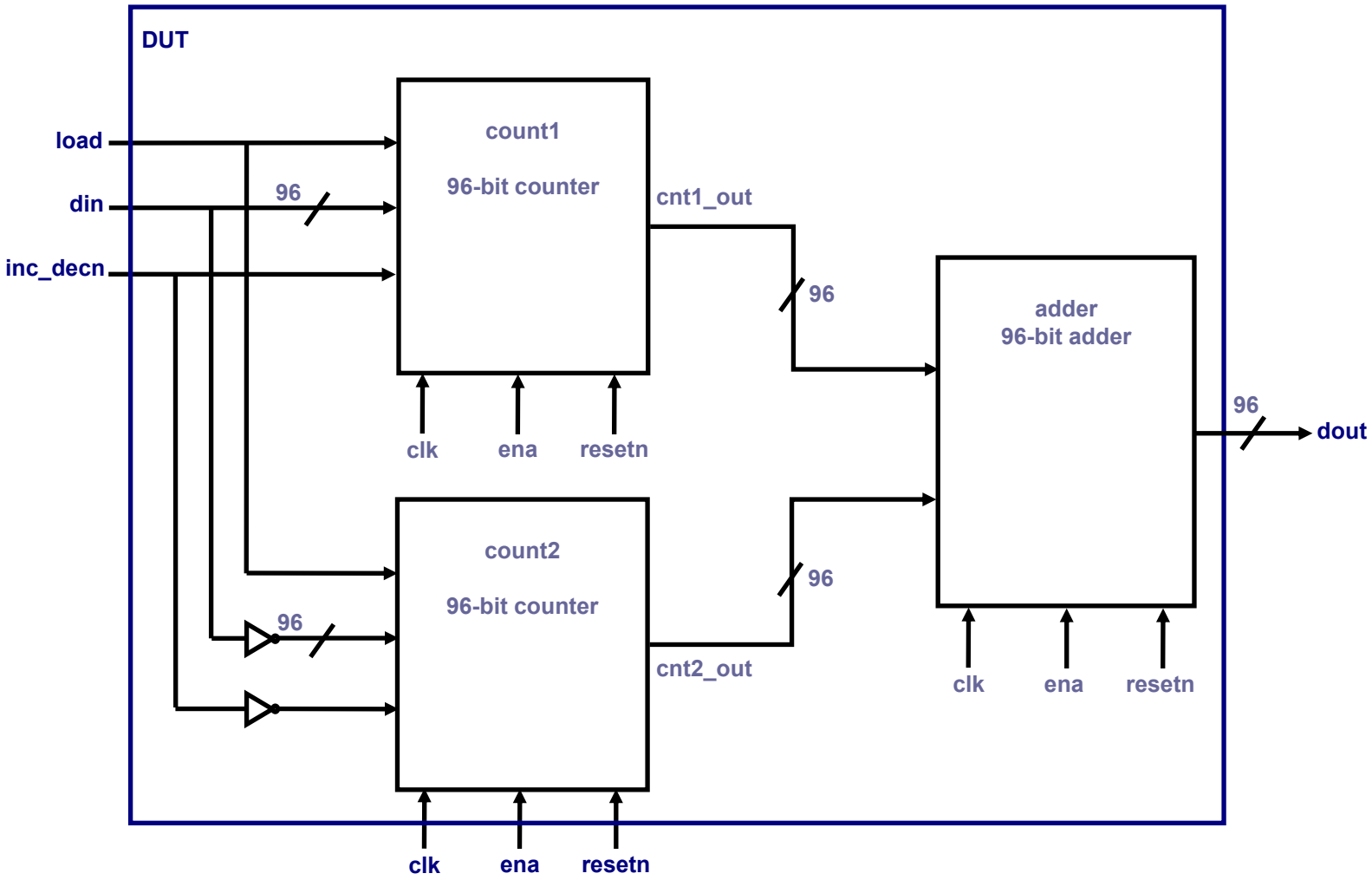
SystemC co-simulation
Lab 1

- In this lab you will learn how to compile and emulate a design for SystemC co-simulation
- First we will compile the design
- Finally we will emulate the design

SystemC co-simulation
Lab 1

```
Trainee/lab1
|-- SystemC_cosim
|   |-- zebu.run
|   |   `-- setup.bash
|   `-- zebu.src
|-- dut
|   |-- probe
|   `-- verilog
|       |-- adder.v
|       |-- counter.v
|       `-- dut.v
`-- testbench
    `-- systemc
        |-- testbench1.cc
        `-- testbench2.cc
```

SystemC co-simulation
Lab 1
The DUT



SystemC co-simulation
Lab 1

- Prepare the ZeBu compilation
  - Go into the `SystemC_cosim` directory
  - Open `zCui`
  - Add `adder.v`, `counter.v` and `dut.v` to the RTL group, define the top module name of the group and select the synthesizer
  - Define the top module name of the design
  - Prepare a probe file in `../dut/probe/probe.tcl`
    - Make sure we declare `dut.count2.cnt[95:0]` as a static probe
    - We also want to see signals `dut.adder.inA` and `dut.adder.inB` as flexible probes
  - Add the probe file to the project

SystemC co-simulation
Lab 1

- Set the target hardware configuration file according to your ZeBu system
- Add the clock in “Clock Declaration” panel
- Save the project
- Generate and modify a DVE file. Static probes need to be manually added to the DVE file but we can generate a template:
  - Left-click on the “Environment” property
  - Click on “Auto Generated”
  - Set “Event-based: HDL Cosimulation”
  - Click on “Generate Now…”
  - If prompted to launch synthesis, click on “Proceed”
  - When the DVE file browser opens, create the DVE file in `zebu.src` directory, name it `lab.dve`
  - (synthesis and DVE generator run)
  - Click on “Edit…”. This opens a text editor on the DVE file
  - Add the static probe in the `output_bin` section of the DVE file

SystemC co-simulation
Lab 1

  - In the “Environment” panel, click on the “Wrapper” tab and set “Wrapper Generation for the Testbench” to “SystemC”
  - Set the job scheduling preferences according to your network settings
- Save the project
- Launch the compilation

SystemC co-simulation
Lab 1

- Run the emulation
  - Go into the `zebu.run` directory
  - Compile the testbench
    ```
    make testbench1
    ```
  - Launch the emulation
    ```
    source setup.bash
    ./testbench1
    ```

HDL co-simulation
Lab 1

- Now add a dynamic probe:
  - Type:
    `zSelectProbes -p ../zcui.work/zebu.work`
  - Select the `dut.count1.cnt[95:0]` signal
  - Save the database
  - Click on the “Generate”>”SystemC” menu item
  - Click on the “Generate”>”Generate” menu item
    - This will generate a new wrapper, have a look at it
  - Exit `zSelectProbes`

SystemC co-simulation
Lab 1

- Run the emulation
  - Go into the `zebu.run` directory
  - Compile the testbench
    ```
    make testbench2
    ```
  - Launch the emulation
    ```
    source setup.bash
    ./testbench2
    ```