Runtime

ZeBu-Server Training

*THE **FASTEST** VERIFICATION*

Copyright ©2010 EVE - Confidential

Agenda

- Selecting signal probes
- Introduction to `zRun`
- Runtime control file

Copyright © 2010 EVE

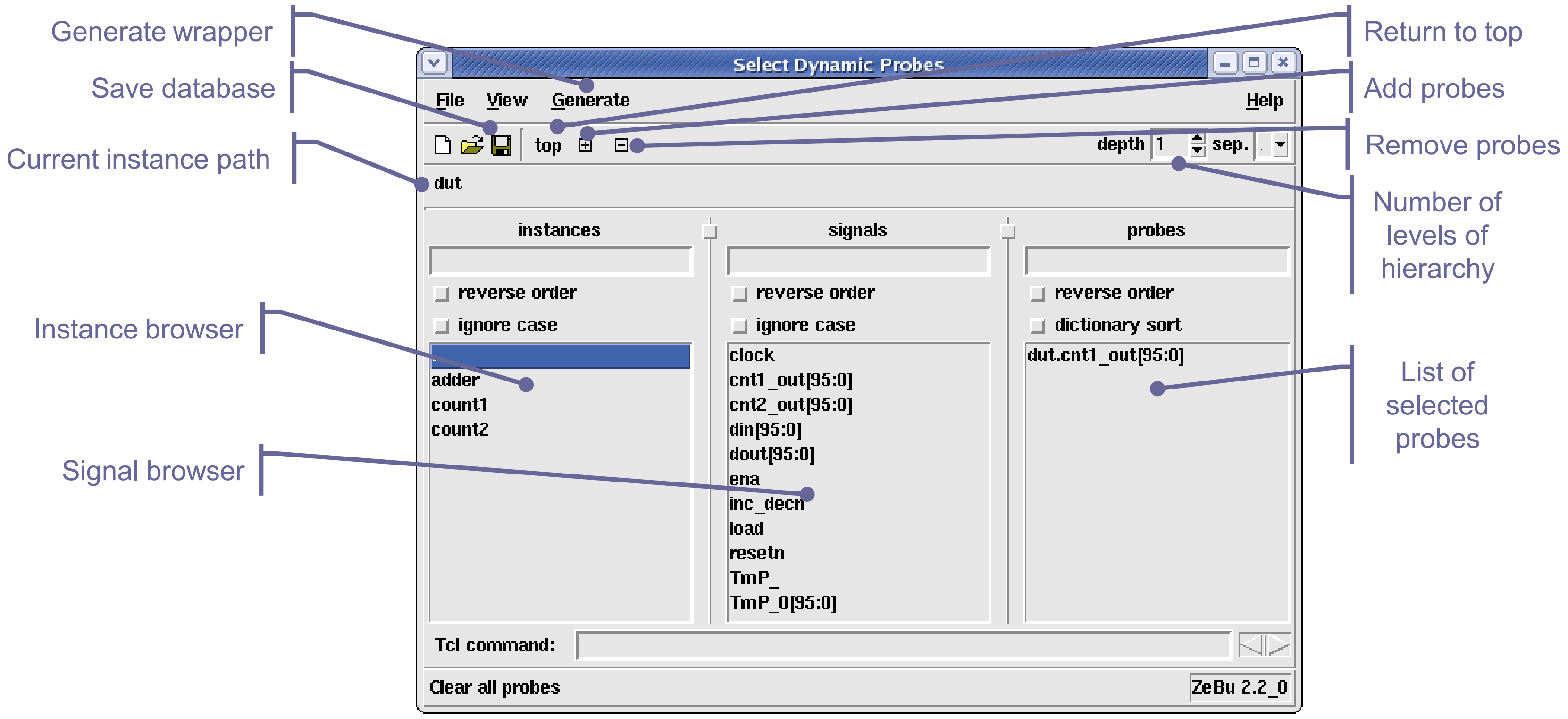
Selecting signal probes

- Dynamic probes and simulated combinational signals are chosen after compilation, at runtime
- They must be "selected" using a ZeBu software utility:
  - Dynamic probes can be selected using `zSelectProbes`
  - Simulated combinational signals must be selected using `zSelectSignals`
- Both tools work the same way
  - Except that `zSelectSignals` requires a database generated using zFAST with appropriate synthesis option
    - Set the accessibility level in the zFAST tab of `zCui` to "Keep all Registers and Simulate Combinational Signals"
  - Other than that, they both have similar command line and graphical user interfaces, use model is the same

Selecting signal probes

- **Type either:**

```
zSelectProbes -p zcui.work/zebu.work
zSelectSignals -p zcui.work/zebu.work
```



Copyright © 2010 EVE

Selecting signal probes

- Use model is:
  - Select dynamic probes (or simulated combinational signals) using the instance/signal browser
  - Save the database (click the save icon)
  - Select the wrapper language (Verilog, VHDL, C++, C or SystemC)
  - Generate the wrapper
  - Run emulation

Copyright © 2010 EVE

Agenda

- Selecting signal probes
- Introduction to `zRun`
- Runtime control file

Runtime Introduction

- Runtime is the phase in which the design is loaded onto the ZeBu-System FPGA and emulation is run
- Runtime has many modes
  - HDL co-simulation
  - C/C++ co-simulation
  - Transactional emulation
  - Synthesizable testbench
  - In Circuit Emulation
- Design must be compiled for the appropriate mode
- In most cases a testbench is used to stimulate and control the design
- A runtime control and debug utility is provided: `zRun`

Copyright © 2010 EVE

Runtime
`zRun` introduction

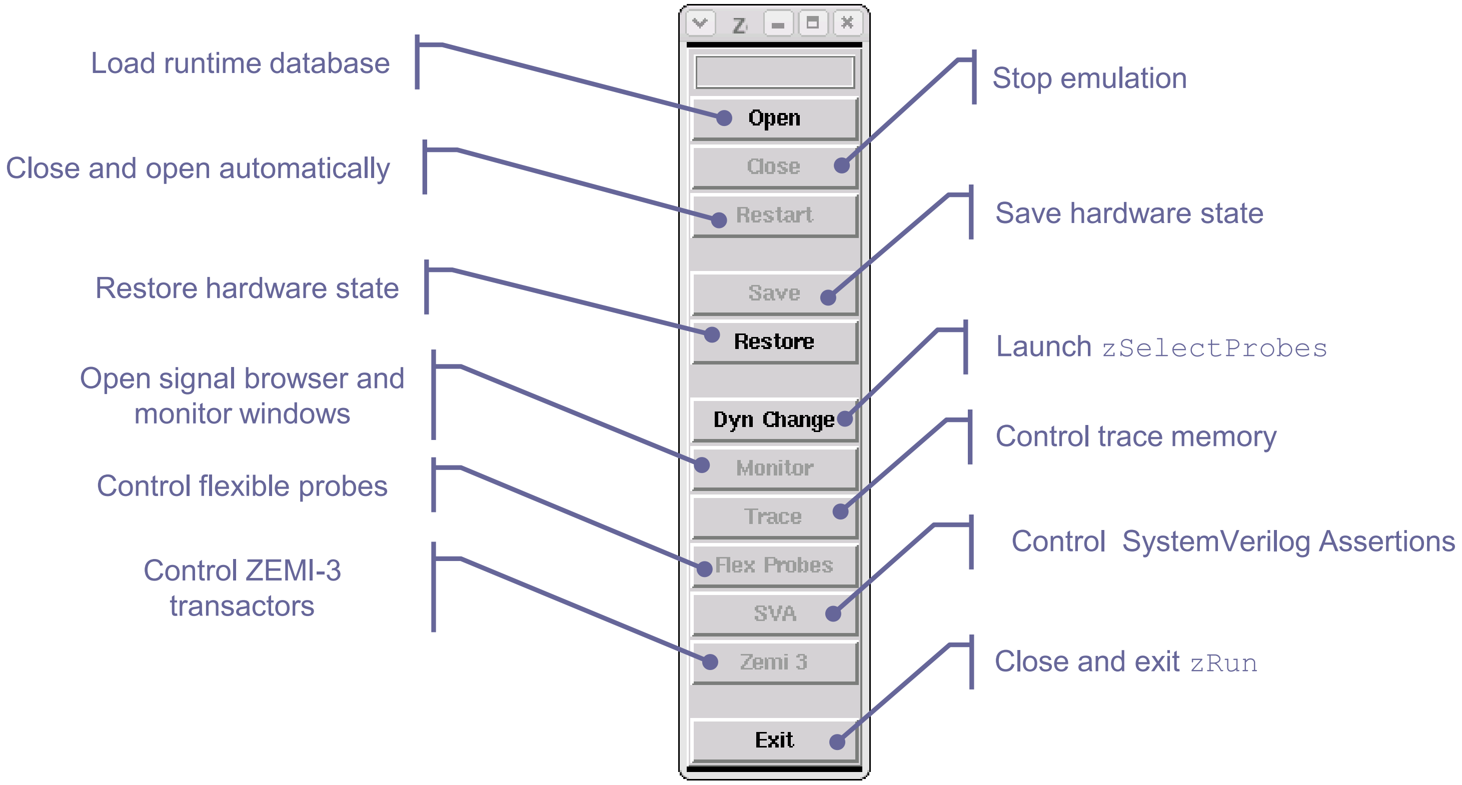
- `zRun` is a TCL/TK application
- It can be completely scripted and/or allows user graphic interaction
- The GUI can be user-defined
- `zRun` provides many debug and control features
- Emulation can, in most cases, run without `zRun` but debug will be less interactive

Runtime
zRun introduction

- **To launch `zRun`, type:**

  ```
  zRun [-testbench <testbench command>] [-zebu.work <path to runtime database>]
  ```

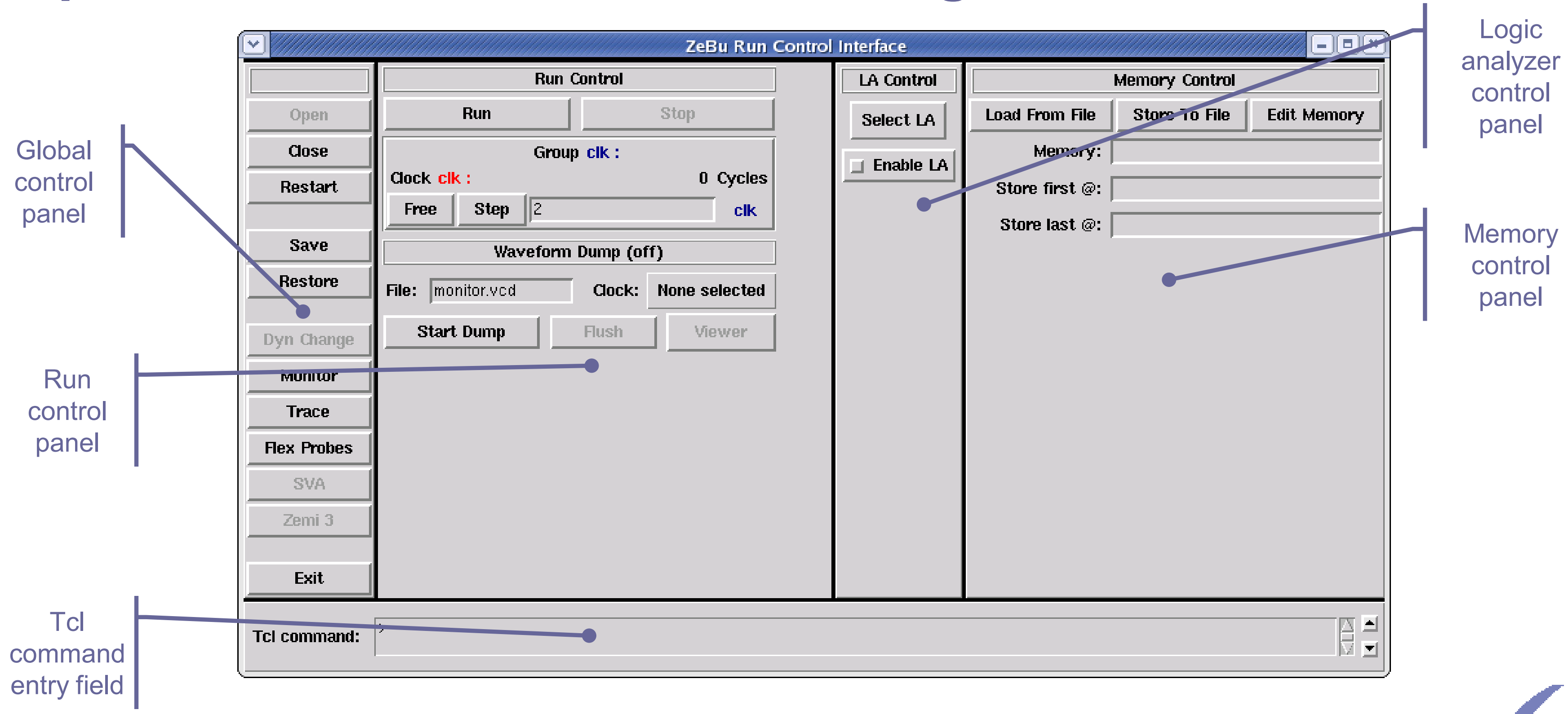
- **This will bring the following window:**



Copyright © 2010 EVE

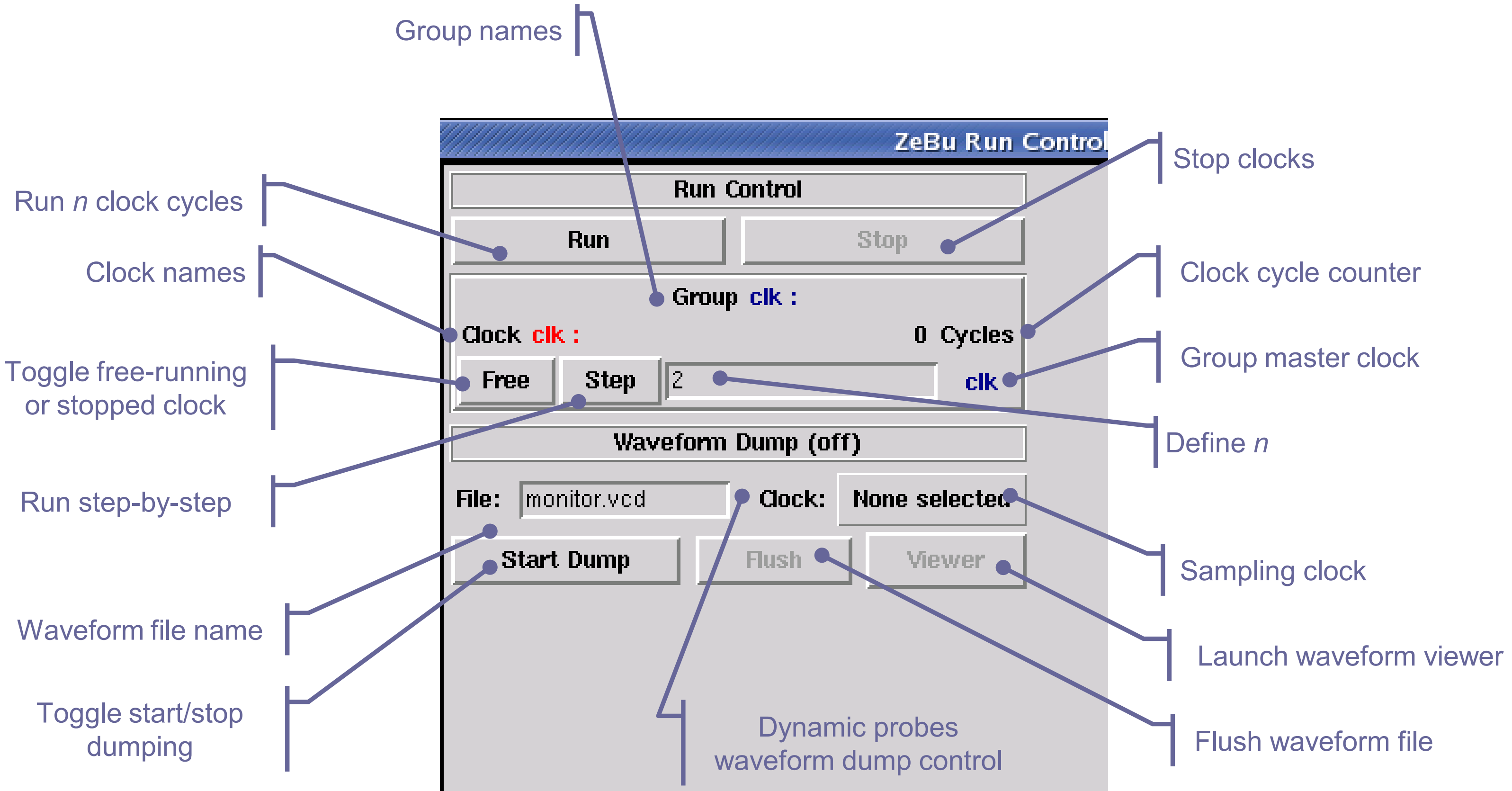
Runtime
zRun introduction

- Once the design is loaded, zRun show these panels:
- Note: the presence of a panel may depend on the presence of a feature in the design



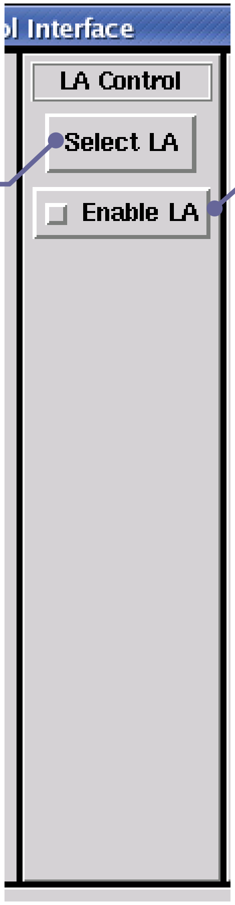
Copyright © 2010 EVE

Runtime
Run Control panel



Copyright © 2010 EVE

Runtime
Logic Analyzer Control panel

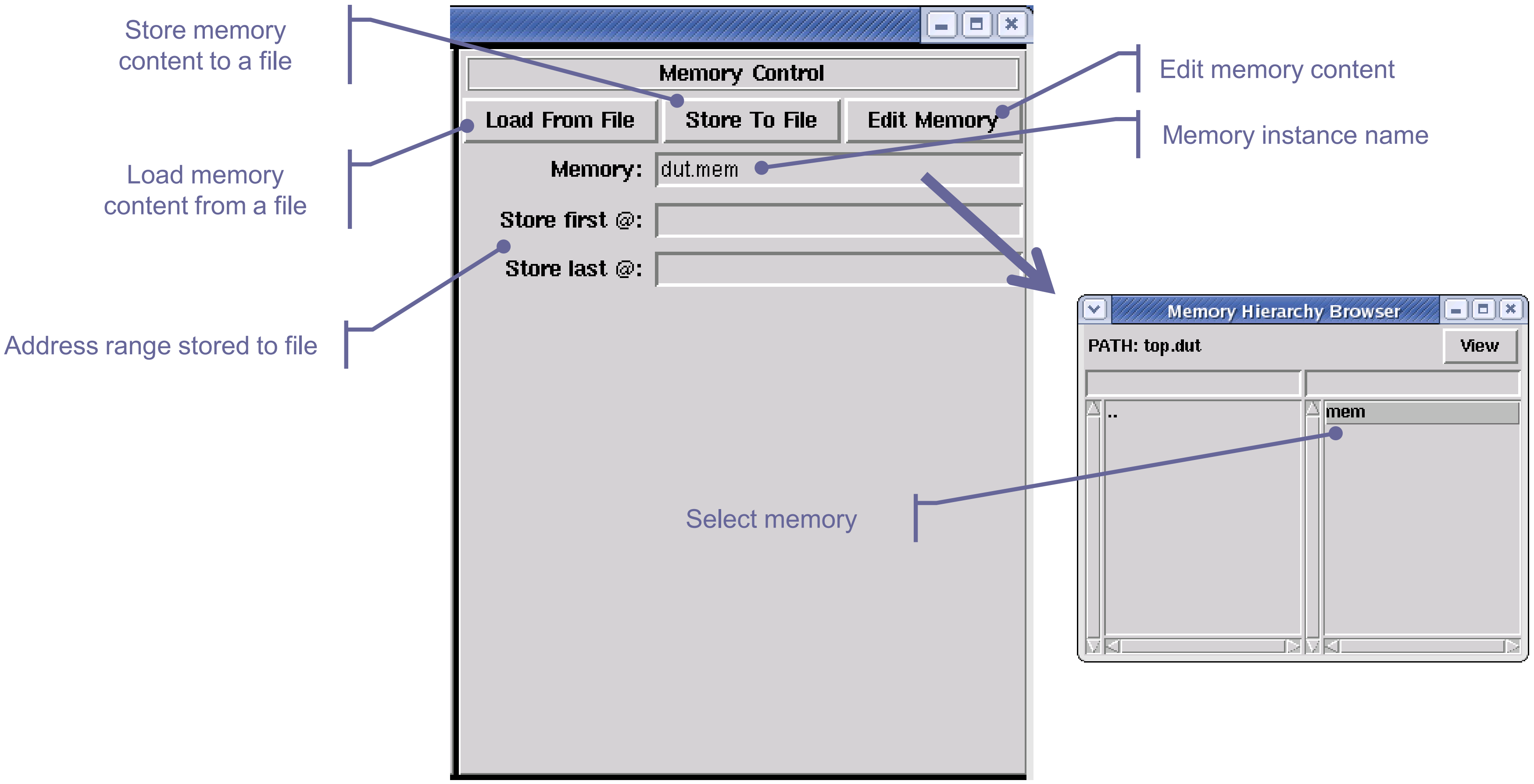


Select which trigger will stop the clock and/or start the trace memory

Enable Logic Analyzer

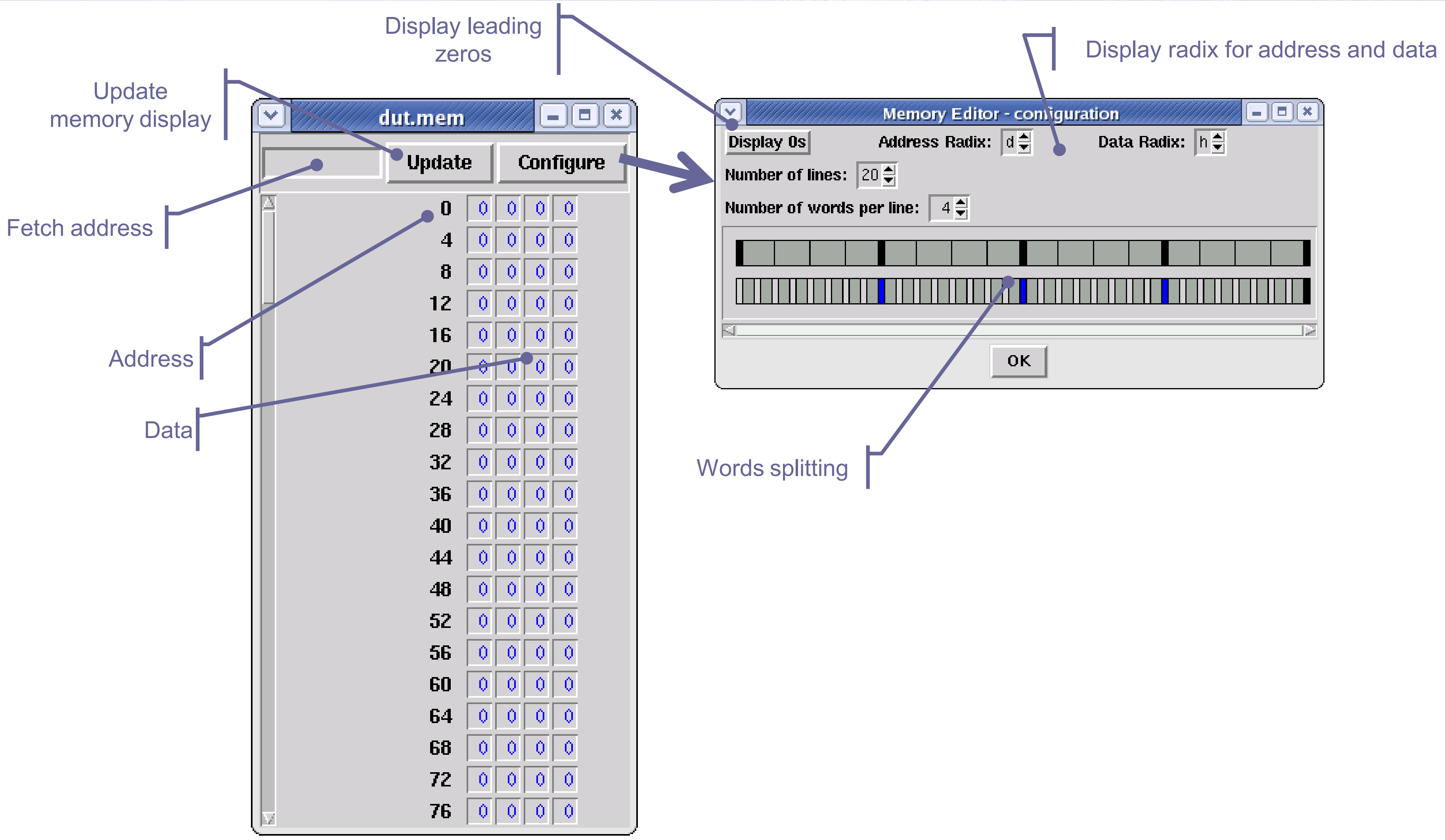
Copyright © 2010 EVE

Runtime
Memory Control panel



Copyright © 2010 EVE

Runtime
Memory editor windows



Display 0s

Address Radix: d

Data Radix: h

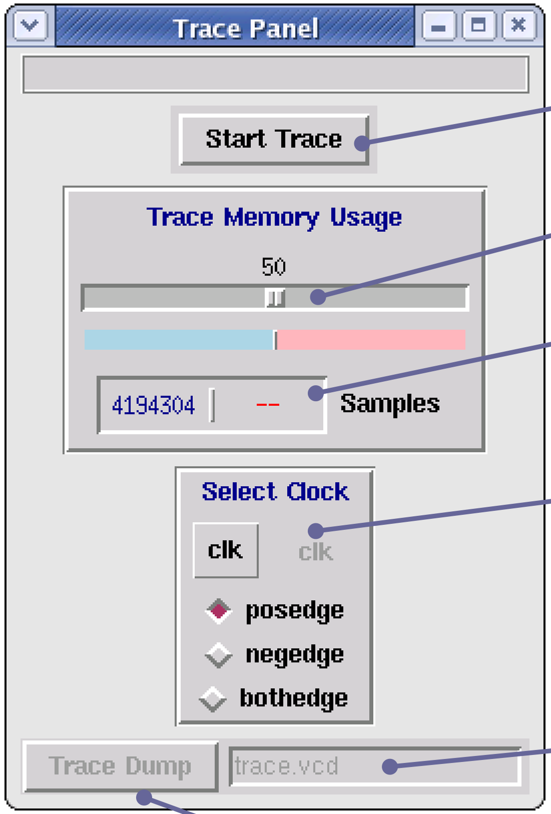
Number of lines: 20

Number of words per line: 4

```
 0  0 0 0 0
 4  0 0 0 0
 8  0 0 0 0
12  0 0 0 0
16  0 0 0 0
20  0 0 0 0
24  0 0 0 0
28  0 0 0 0
32  0 0 0 0
36  0 0 0 0
40  0 0 0 0
44  0 0 0 0
48  0 0 0 0
52  0 0 0 0
56  0 0 0 0
60  0 0 0 0
64  0 0 0 0
68  0 0 0 0
72  0 0 0 0
76  0 0 0 0
```

Copyright © 2010 EVE

Runtime Trace Panel



Trace Panel

Start Trace

Trace Memory Usage

50

4194304 | -- Samples

Select Clock

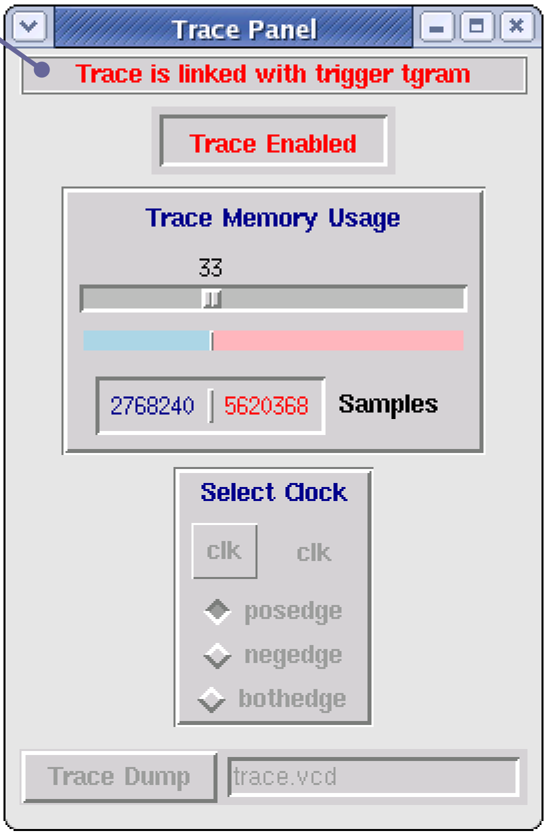
clk clk

posedge

negedge

bothedge

Trace Dump trace.vcd



Trace Panel

Trace is linked with trigger tgram

Trace Enabled

Trace Memory Usage

33

2768240 | 5620368 Samples

Select Clock

clk clk

posedge

negedge

bothedge

Trace Dump trace.vcd

Trace memory is linked to a trigger

Toggle enable trace memory

Pre-trigger ratio (%)

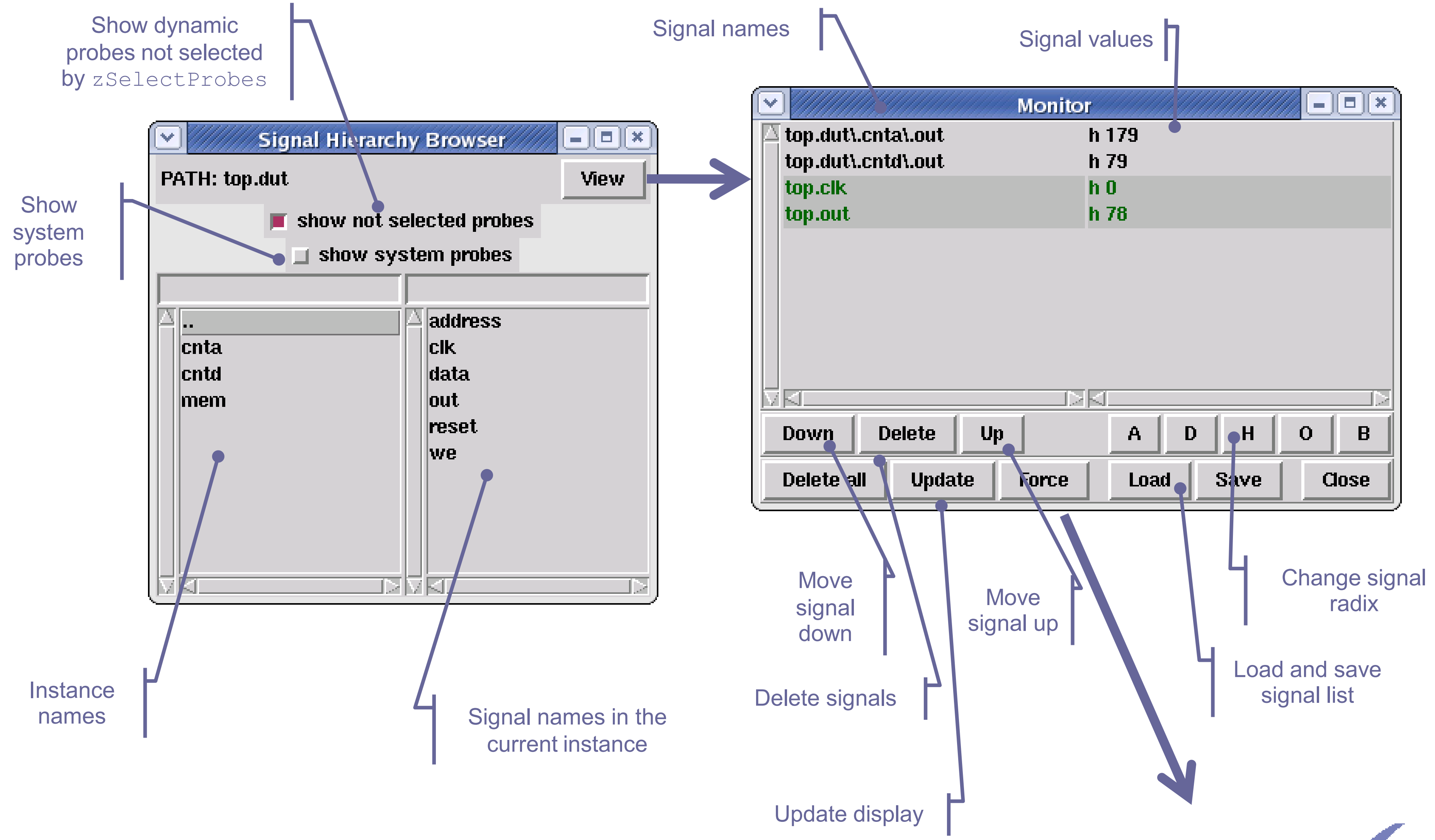
Number of sample captured

Select sampling clock

Trace dump file name

Dump waveform to trace file

Runtime
Signal Hierarchy Browser and Monitor panels



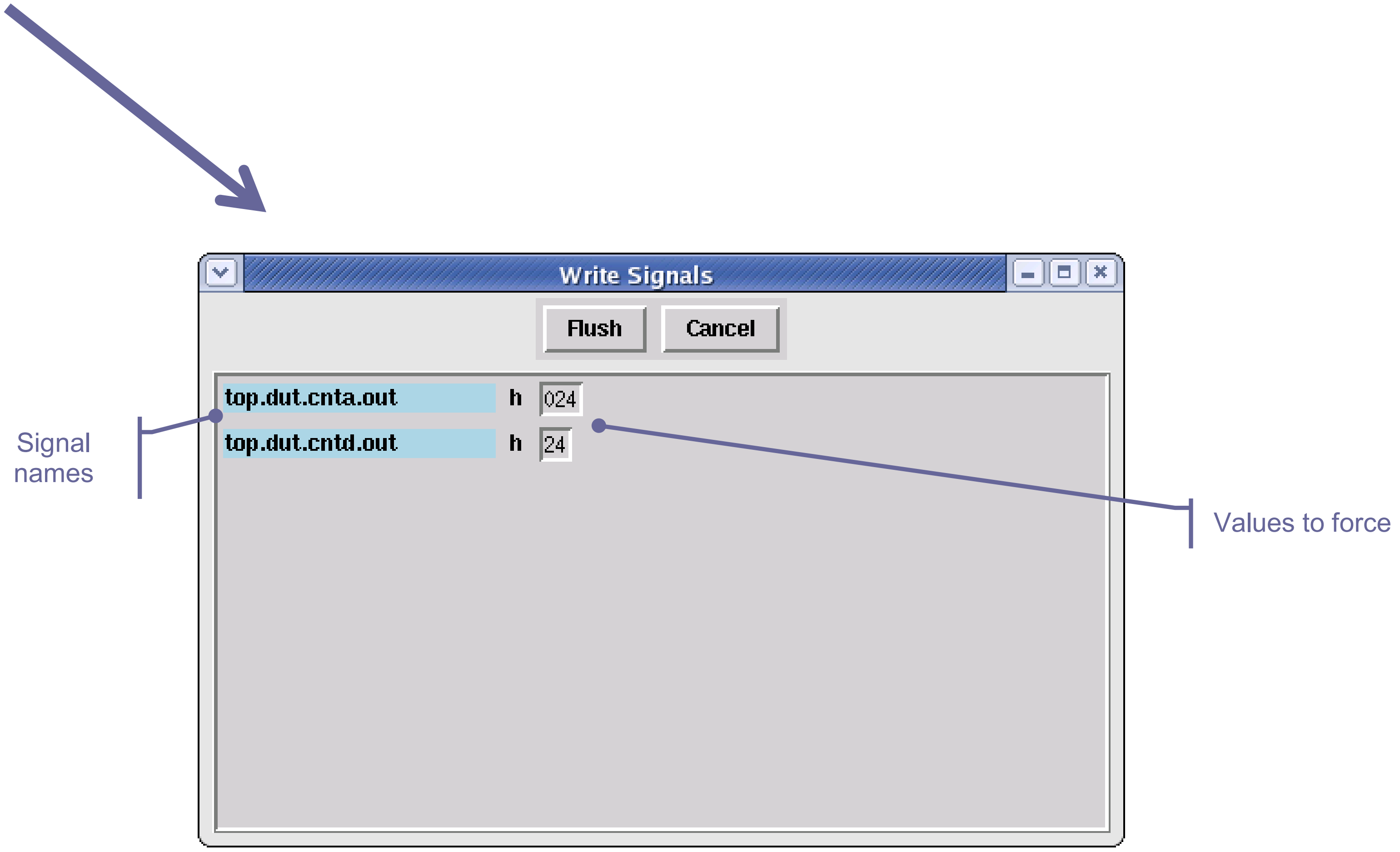
PATH: top.dut

..
cnta
cntd
mem

address
clk
data
out
reset
we

```
top.dut\.cnta\.out    h 179
top.dut\.cntd\.out    h 79
top.clk               h 0
top.out               h 78
```

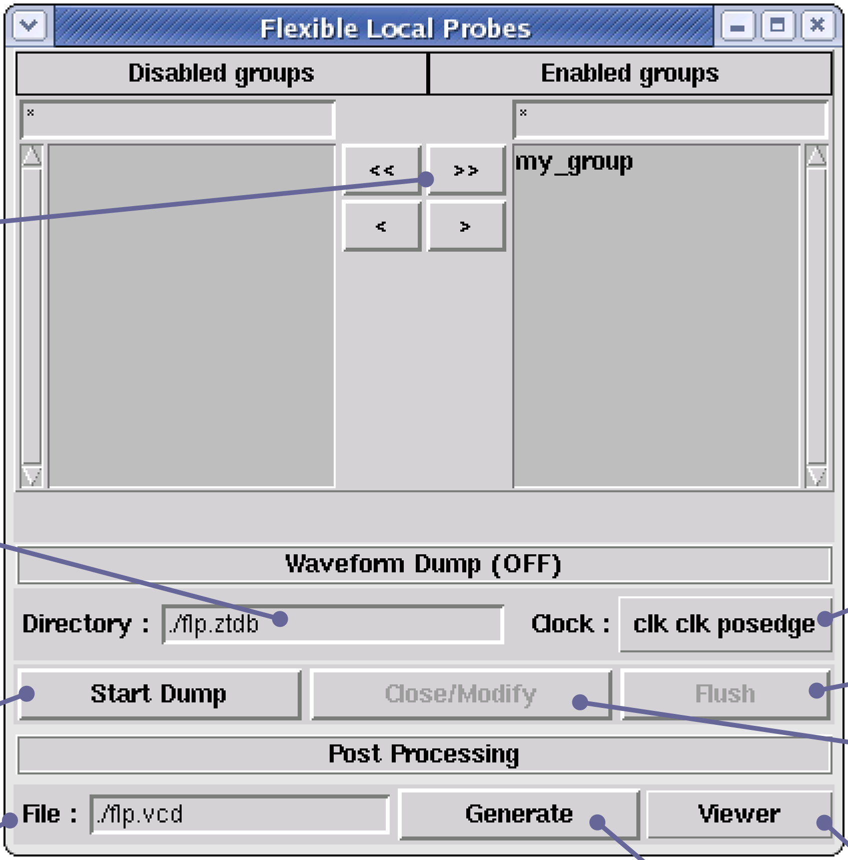
Runtime Write Signals panel



top.dut.cnta.out	h	024
top.dut.cntd.out	h	24

Copyright © 2010 EVE

Runtime
Flexible Local Probes panel



Select probe groups which are enabled

Name of directory where intermediate raw format trace file will be stored

Toggle to activate the trace dump

Final waveform file name

Select sampling clock

Flush data to disk

Close trace file

Launch waveform viewer

Generate waveform file from trace file

Copyright © 2010 EVE

Runtime
`zRun` lab 1

- In this lab you will learn how to use `zRun` by practicing most menus and buttons
- But first you need to compile the design

Copyright © 2010 EVE

Runtime
zRun lab 1

```
Trainee/lab1
|-- dut
|   `-- verilog
|       |-- adder.v
|       |-- counter.v
|       `-- dut.v
`-- zRun
    |-- project.zpf
    |-- testbench
    |   `-- testbench.cc
    |-- zebu.run
    |   `-- Makefile
    `-- zebu.src
        `-- test.dve
```

Copyright © 2010 EVE

Runtime
`zRun` lab 1

- Go into the `zRun` directory and load the `project.zpf` file into `zCui`:
  `zCui –p project.zpf`
- Select the target hardware configuration file
  - Left-click on the “Back-End : default” property
  - Open the file browser and select the appropriate target hardware configuration file for your ZeBu system
- Set the job scheduling preferences
  - Left-click on the “Job Scheduling” property
  - Fill the form in accordance with your network configuration
- Save the project
- Launch the compilation
- Once finished exit `zCui`

Runtime
`zRun` lab 1

- Go into the `zebu.run` directory and build the testbench executable:

  ```
  cd zebu.run
  make
  ```

- Then launch `zRun` and the testbench:

  ```
  zRun -testbench ./testbench
  ```

Copyright © 2010 EVE

Runtime
`zRun` lab 1

- Click on “Open”
- Run full co-simulation
  - Click on “Free”
  - Check that the clock counter is at 10,000,005
- Click on “Restart”

Runtime
`zRun` lab 1

- Click on “Monitor”
- Check “show not selected probes”
- Double-click “`dut`”, select all signals from “clock” to “`resetn`”
- Click on “View”
- Click about 10 times on “Step” to run step-by-step, see how monitored values are changing
- Click on “Free” then on “Update” and see how monitored values are updated while the design is running
- Click on “Restart”

Copyright © 2010 EVE

Runtime
`zRun` lab 1

- Make sure signals are still visible in the monitor window
  - Only these signals are dumped
- Select "`clock`" as dump clock
- Click on "Start Dump"
- Click on "Free", wait for about 10,000 cycles and click on "Stop"
- Then click on "Stop Dump" and then on "Close"
- Look at the `monitor.vcd` file in your favorite waveform viewer
- Click on "Restart"

Copyright © 2010 EVE

Runtime
`zRun` lab 1

- Click on “Select LA > Trace on Trig > tgram”
- In the “tgram expression” window, type the following expression:
  - `dut.count1.cnt[31:0]==32'hbabe0100`
  - Click on “Ok”
- Click on “Trace Enabled” to switch it off
- Set the “Trace Memory Usage” to zero
- Select “`clock`” as trace clock (clock group named “`myGroup`”)
- Click on “Start Trace” to switch it on
- Click on “Free”
- Click on “Stop” when trace capture is completed
- Click on “Trace Enabled” to switch it off
- Click on “Trace Dump”
- Look at the `trace.vcd` file in your favorite waveform viewer
- Click on “Restart”

Copyright © 2010 EVE

Runtime
`zRun` lab 1

- Click on Flex Probes
- Select “`thegroup`” from the “Disabled groups” panel
- Click on the single right arrow button to enable the group
- Select the posedge of “`clock`” as the sampling clock
- Click on “Start Dump”
- Click on “Free”, then on “Stop” when the counter reaches 10,000,005
- Click on “Stop Dump”
- Click on “Close/Modify”
- Click on “Generate”
- Look at the `flp.vcd` file in your favorite waveform viewer
- Click on “Exit”

Copyright © 2010 EVE

Agenda

- Selecting signal probes
- Introduction to `zRun`
- Runtime control file

Copyright © 2010 EVE

Runtime Configuration (`designFeatures`)

- If a file called `designFeatures` is present in the current runtime directory, it will be used to setup all kinds of runtime configurable options:
  - Setting primary clock ratios and phase relationship
  - Setting emulation frequency
  - Initializing memories at time 0 from files
- Check out `designFeatures.help` file for detailed syntax

Copyright © 2010 EVE

Runtime `designFeatures`: Primary clocks

- Each primary clock (driven by a `zceiClockPort` in the DVE) can be defined as:
  - Controlled
    The clock is generated by the ZeBu clock generator and has guaranteed phase and ratio with respect to other primary clocks
  - Free-running
    The clock is generated by the ZeBu clock generator at a fixed frequency
  - In-Situ
    The clock is driven from an external target, you can't specify a frequency

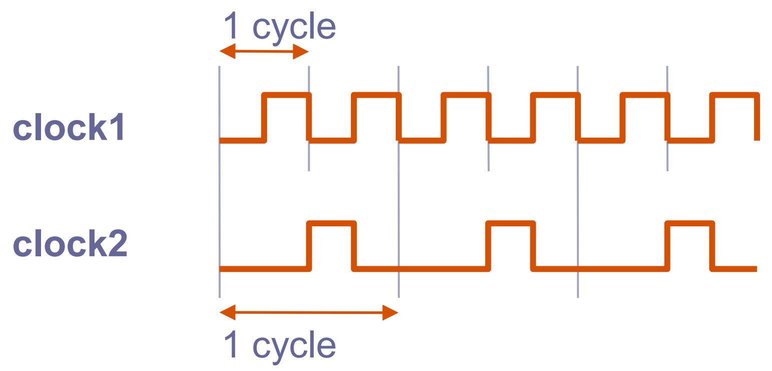
Runtime designFeatures: Controlled clocks

```
$U0.M0.my_clock.Mode = "controlled";
$U0.M0.my_clock.Waveform = "_-";
$U0.M0.my_clock.VirtualFrequency = 366; # 366 MHz virtual
$U0.M0.my_clock.GroupName = "corePLL";
```

Copyright © 2010 EVE

Runtime
designFeatures: Controlled clocks

```
$U0.M0.clock1.Mode = "controlled";
$U0.M0.clock1.GroupName = "Group1";
$U0.M0.clock1.Waveform = "_-";
$U0.M0.clock1.VirtualFrequency = 100;
$U0.M0.clock2.Mode = "controlled";
$U0.M0.clock2.GroupName = "Group1";
$U0.M0.clock2.Waveform = "__-_";
$U0.M0.clock2.VirtualFrequency = 50;
```



- 100/50 = 2, clock1 is twice faster than clock2
- The clocks belonging to the same group run together

Copyright © 2010 EVE

Runtime `designFeatures`: Free-running clocks

```
$U0.M0.my_clock.Mode = "free-running";
$U0.M0.my_clock.Frequency = 5000; # 5 MHz real
```

Copyright © 2010 EVE

Runtime designFeatures: In-situ clocks

```
$U0.M0.my_clock.Mode = "in-situ";
```

Runtime `designFeatures`: emulation frequency

- `driverClock` is the key frequency that determines the actual emulation speed of controlled clocks
  - At every cycle of `driverClock`, the next edge of the controlled clocks is generated
  - In simple cases, this means that the real DUT clock frequency is half the `driverClock` frequency

```
$driverClk.Frequency = 4000;  # 4 MHz
```

Copyright © 2010 EVE

Runtime `designFeatures:` zDelay

- zDelay is the amount of time during which the data path is frozen to let the clock tree propagate
  - Delay is expressed in real time (ns)

```
$zClockSkewTime = 400;          # 400 ns
```

  - Delay cannot be larger than period of `driverClock`, or design will not be functional

Copyright © 2010 EVE

Runtime `designFeatures`: Filter delay

- The filter time is the amount of time necessairy to let the clock tree propagate
  - Delay is expressed in real time (ns)

```
$zClockFilterTime=200;          # 200 ns
```

  - Delay cannot be larger than `zClockSkewTime`, else the design will not be functional
  - Used only in filter glitch mode

Copyright © 2010 EVE

Runtime `designFeatures`: Initializing memories

- All BRAMs and ZRMs can be initialized at time 0:
  - **`designFeatures`:**

```
$memoryInitDB = "memory.init";
```

  - **`memory.init`:**

```
top.sram.mem0          ./images/mem0.hex
top.sram.mem1          ./images/mem1.hex
top.cpu.icache         ./caches/zero.hex
```