Getting Started with Veloce

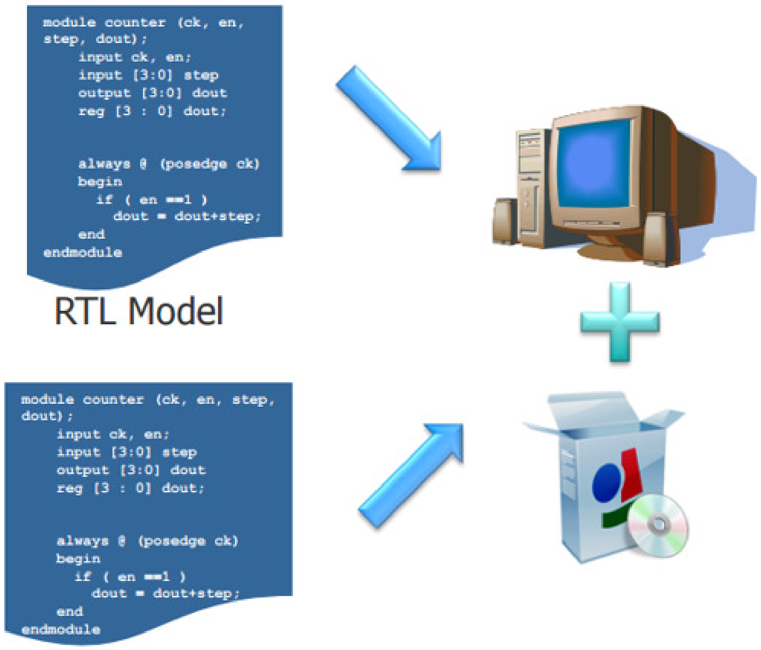
Outline:

- Why Emulate?
- Veloce Overview
- PSU Veloce Environment
- Veloce Compile Flow
- Veloce Use Modes
  - ICE/Standalone Mode
  - ~~TBX/Comodel Mode~~
- Summary
- ~~Veloce Strato~~
- Need for Speed Emulation Contest

WHY EMULATE?

Software Simulation

HDL – Verilog, SystemVerilog, VHDL



```
module counter (ck, en,
step, dout);
    input ck, en;
    input [3:0] step
    output [3:0] dout
    reg [3 : 0] dout;


    always @ (posedge ck)
    begin
      if ( en ==1 )
        dout = dout+step;
    end
endmodule
```

RTL Model

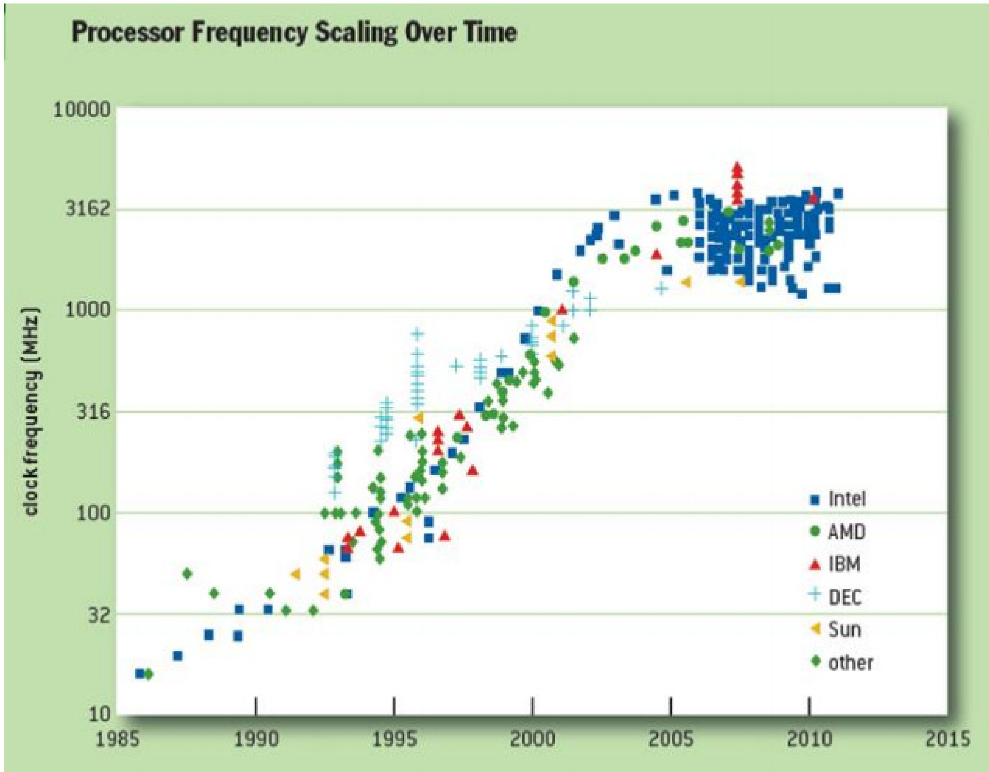
```
module counter (ck, en, step,
dout);
    input ck, en;
    input [3:0] step
    output [3:0] dout
    reg [3 : 0] dout;


    always @ (posedge ck)
    begin
      if ( en ==1 )
        dout = dout+step;
    end
endmodule
```

Test bench

HVL – SystemVerilog, SystemC, C

- Model the functionality in HDL
- Simulator software executes HDL to ‘simulate’ hardware parallelism
- Testbench produces test vectors
- Simulator runs on an underlying processor
- Mostly sequential execution

Processor Clock Speed Scaling Stalls



Source: Recording Microprocessor History 4/6/2012 Andrew Danowitz, Kyle Kelley, James Mao, John P. Stevenson, Mark Horowitz
http://queue.acm.org/detail.cfm?id=2181798

- Raw processor clock frequencies have stalled
- Performance of simulator = f (CPU frequency)
  - Mostly sequential execution
- Hardware Emulation accelerates design frequency
- Typical Software Simulator
  - Effective Frequency in range of Hz to few KHz
- Mentor Graphics Veloce2 Emulator
  - Frequency up to 2MHz !

VELOCE OVERVIEW

Veloce Overview

- Veloce : (Italian ve-law-che) Rapid, Fast
- Veloce : The Best Emulation Architecture

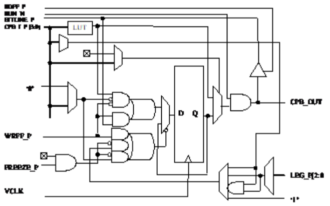
- Fast compile
- High capacity
- High performance
- Full visibility
- Power efficient

Veloce Overview

RTL Design

```
module ddr1_core (DOUT, DIN, WA, RA, WE);
    input [23 : 0] WA, RA;
    input [7 : 0] DIN;
    input WE;
    output [7 : 0] DOUT;
    reg [7 : 0] DOUT;
    reg [7 : 0] mem [16777215 : 0];

    always @ (posedge WE)
    begin
        mem[WA] = DIN;
    end

    always @ (RA)
    begin
        DOUT = mem[RA];
    end
 endmodule
```

Compiler



Synthesis

Crystal

16 CRY

AVB

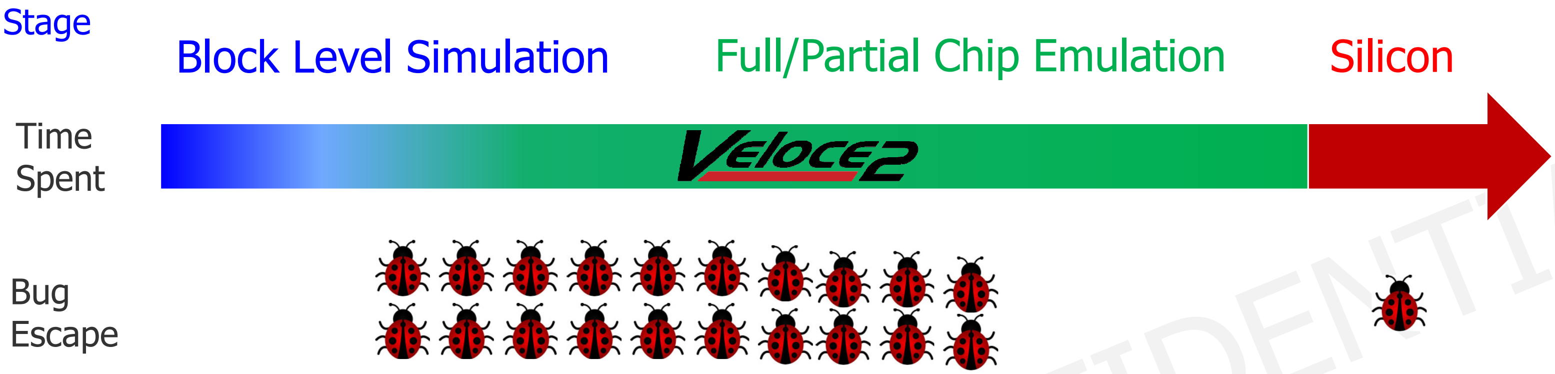
16 AVBs

Veloce2 Quattro

Verification Flow **without** Veloce

Stage	Block Level Simulation	Full Chip Simulation	Silicon in Lab
Time Spent	80%	10%	10 %
Bug Escape	🐞🐞🐞🐞🐞🐞🐞 🐞🐞🐞🐞🐞🐞🐞	🐞🐞🐞 🐞🐞🐞	🐞🐞🐞
Correction Cost	Low	High	Extremely High
Debug	Efficient	Efficient	Extremely Difficult
Turn Around	Fast and Easy	Slow	Re-spin!

Verification Using Veloce



Stage

Block Level Simulation

Full/Partial Chip Emulation

Silicon

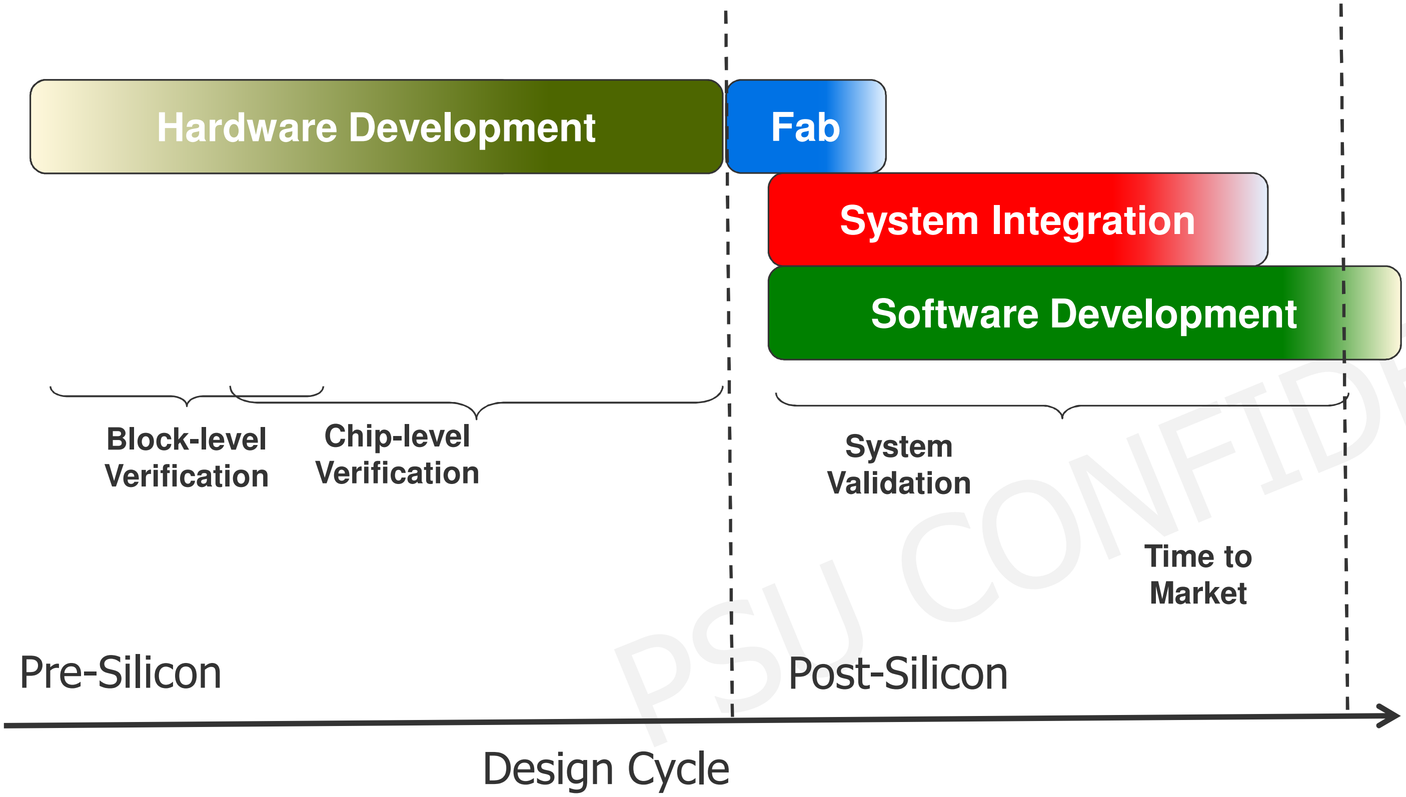
Time Spent

Veloce2

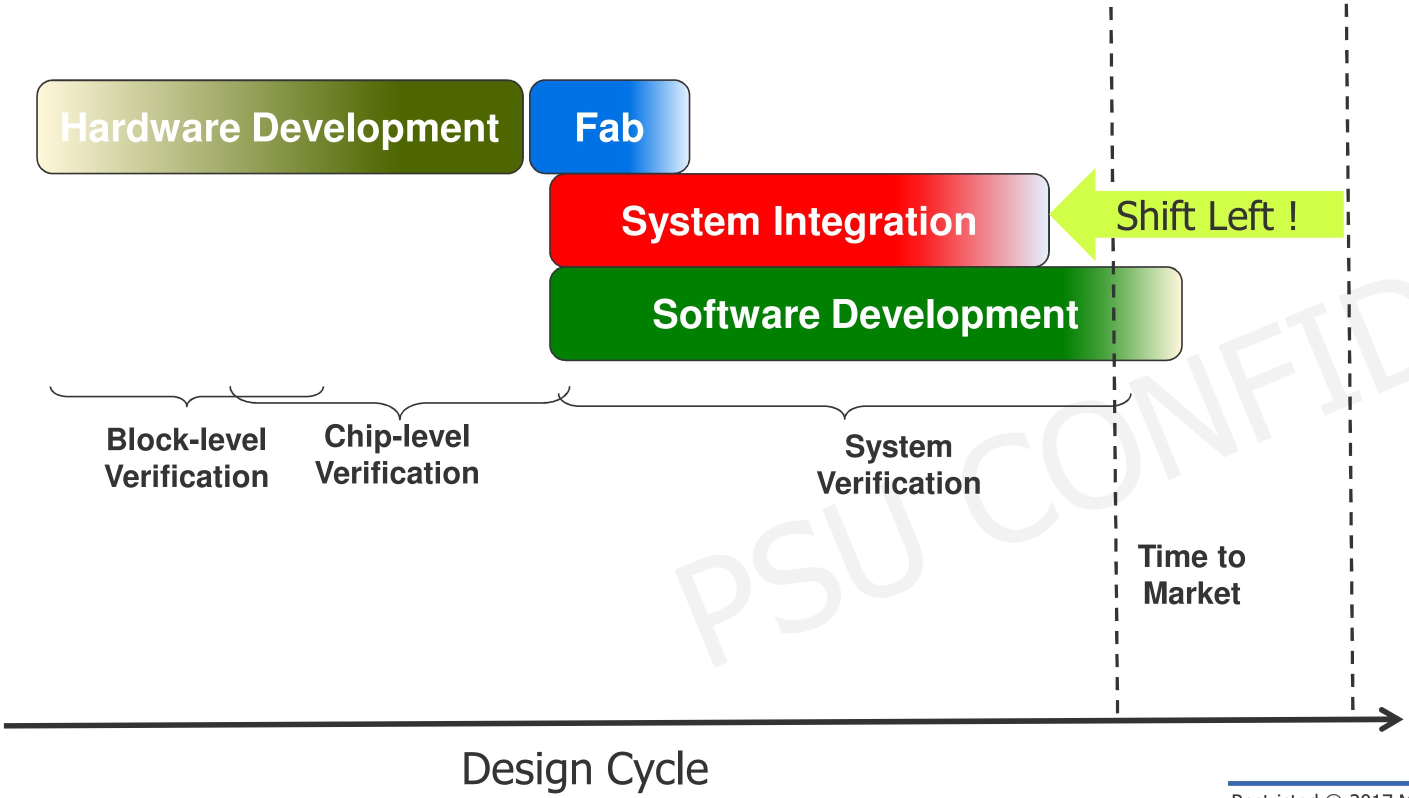
Bug Escape

Veloce2	
Bug Correction Cost	Low
Debug	Efficient
Turn around	Fast

System Development Cycle



System Development Cycle *with* Veloce



Hardware Development

Fab

System Integration

Shift Left !

Software Development

Block-level Verification

Chip-level Verification

System Verification

Time to Market

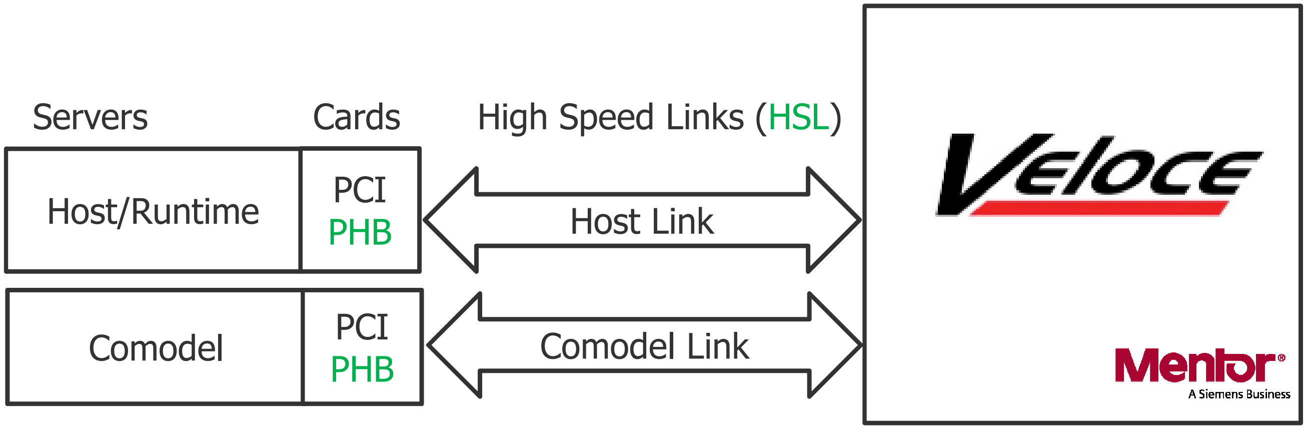
Design Cycle

Mentor®
A Siemens Business

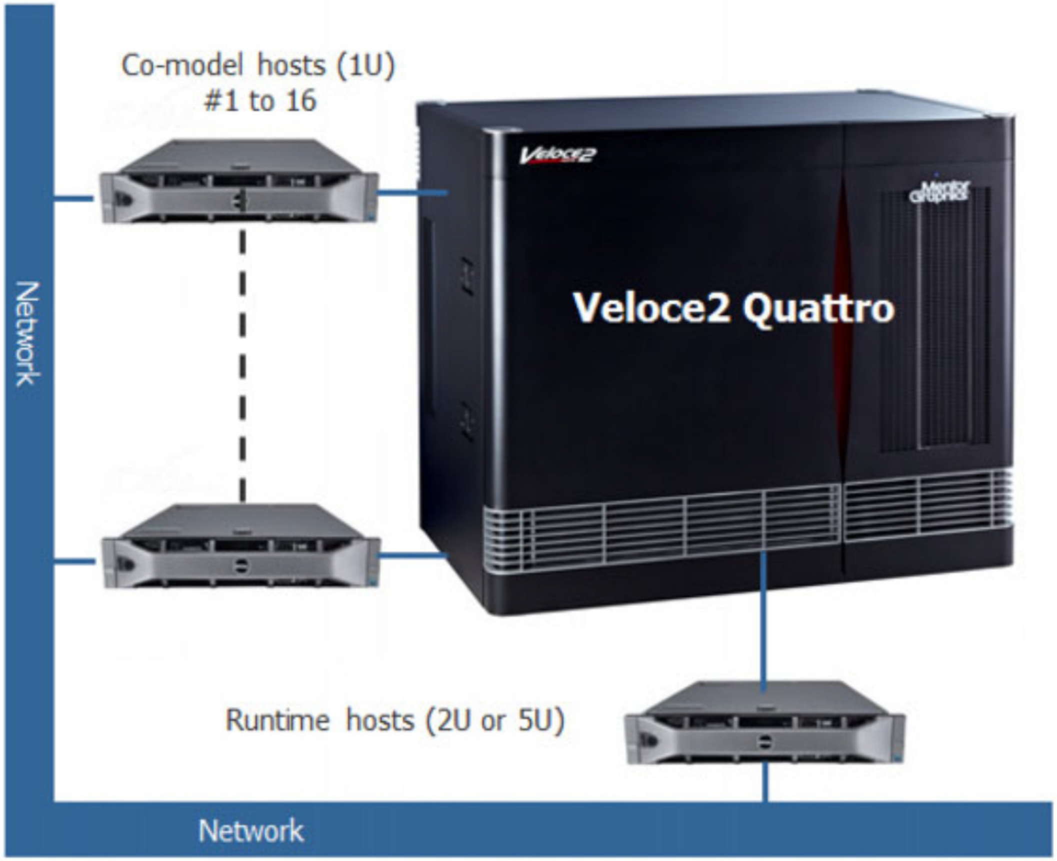
PSU VELOCE ENVIRONMENT

Overview of Veloce Emulation Environment

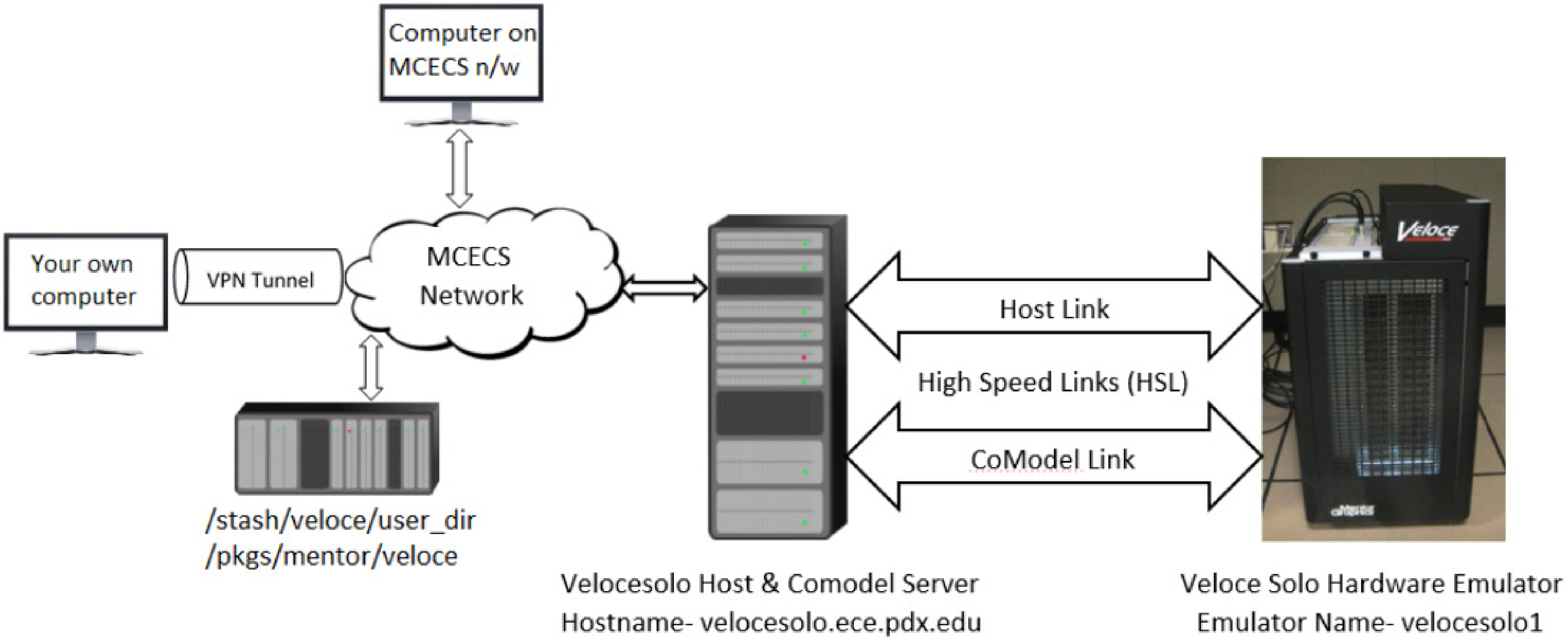
- Platform Host Board (PHB) – PCI Card installed in server for connection to HSL
- High Speed Link (HSL) – Proprietary High Speed Cable
- Host Link – Primary connection, used for design download and trace upload
- Comodel Link – Used for carrying transaction (testbench) data



Veloce Emulation Environment – Quattro Example



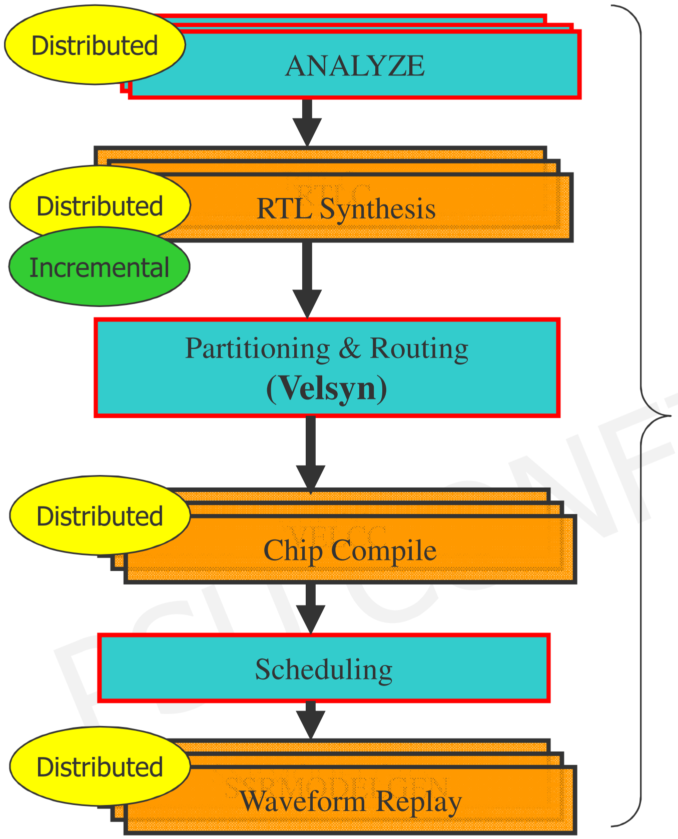
Veloce Emulation Environment – PSU Setup



COMPILE FLOW

Compile Flow

- Complete flow developed by mentor
  - No dependence on FPGA vendor SW
  - Fast turnaround on compile issues
  - Encryption
- Continuously enhanced
  - Capacity utilization
  - Runtime speed
  - Compile speed
  - Language support



Distributed

ANALYZE

Distributed

Incremental

RTLC

RTL Synthesis

Partitioning & Routing

(Velsyn)

Distributed

VELCC

Chip Compile

Scheduling

Distributed

SSRMODELGEN

Waveform Replay

40-80M gates per hour

Multi Step Compilation Flow

Create config file (Veloce.config)

```
vellib work
velmap work work
velanalyze –hdl verilog <RTL Source>
velanalyze –hdl vhdl <RTL Source>

velcomp -top my_top
```

veloce.config

- `veloce.config` will contain options for analysis and compile
- All parameters need to be specified before compilation, can not be changed during compile
- All runtime parameters need to be specified in a separate file called velrunopts.ini which is the runtime initialization file
- The different phase names appearing in the veloce.config file

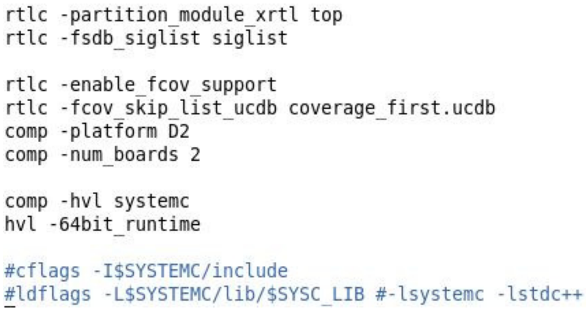
comp
veanalyze/vhanalyze
rtlc
velsyn

velcc
velgs
hvl
source

veloce.config …

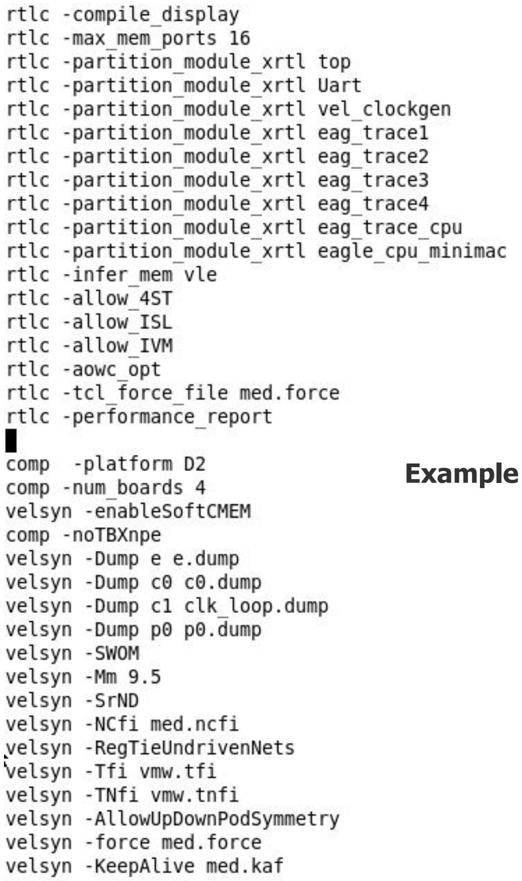
- Configuration file for the Veloce emulation model
- Follows the following semantics:
  - Phase specific options given with name of phase
    - “rtlc -<option>”, “velsyn -<option>”, “hvl -<option>”, etc.
  - Multiple lines for specifying options in same phase supported
    - comp -platform D1
    - comp -num_boards 2
  - Options applicable to multiple phases given with “comp”
    - comp –cui_file vmw.cui
- Different name cannot be used for the config file
  - Only use veloce.config

veloce.config Examples



```
rtlc -partition_module_xrtl top
rtlc -fsdb_siglist siglist

rtlc -enable_fcov_support
rtlc -fcov_skip_list_ucdb coverage_first.ucdb
comp -platform D2
comp -num_boards 2

comp -hvl systemc
hvl -64bit_runtime

#cflags -I$SYSTEMC/include
#ldflags -L$SYSTEMC/lib/$SYSC_LIB #-lsystemc -lstdc++
```

Example 1



```
rtlc -compile_display
rtlc -max_mem_ports 16
rtlc -partition_module_xrtl top
rtlc -partition_module_xrtl Uart
rtlc -partition_module_xrtl vel_clockgen
rtlc -partition_module_xrtl eag_trace1
rtlc -partition_module_xrtl eag_trace2
rtlc -partition_module_xrtl eag_trace3
rtlc -partition_module_xrtl eag_trace4
rtlc -partition_module_xrtl eag_trace_cpu
rtlc -partition_module_xrtl eagle_cpu_minimac
rtlc -infer_mem vle
rtlc -allow_4ST
rtlc -allow_ISL
rtlc -allow_IVM
rtlc -aowc_opt
rtlc -tcl_force_file med.force
rtlc -performance_report

comp  -platform D2
comp -num_boards 4
velsyn -enableSoftCMEM
comp -noTBXnpe
velsyn -Dump e e.dump
velsyn -Dump c0 c0.dump
velsyn -Dump c1 clk_loop.dump
velsyn -Dump p0 p0.dump
velsyn -SWOM
velsyn -Mm 9.5
velsyn -SrND
velsyn -NCfi med.ncfi
velsyn -RegTieUndrivenNets
velsyn -Tfi vmw.tfi
velsyn -TNfi vmw.tnfi
velsyn -AllowUpDownPodSymmetry
velsyn -force med.force
velsyn -KeepAlive med.kaf
```

Example 2

velcomp

- “velcomp” is main compile engine command for all tasks
  - velcomp runs the full compile from analysis to ssrgen
- Sub tasks
  - velcomp –task <task name> < analyze,rtlc,velsyn,velcc,…..>
- Starting from sub task
  - velcomp –start_task <task name>
- Ending till sub task
  - velcomp –end_task <task name>
- One task to another
  - velcomp –start_task <task 1> -end_task <task 2>

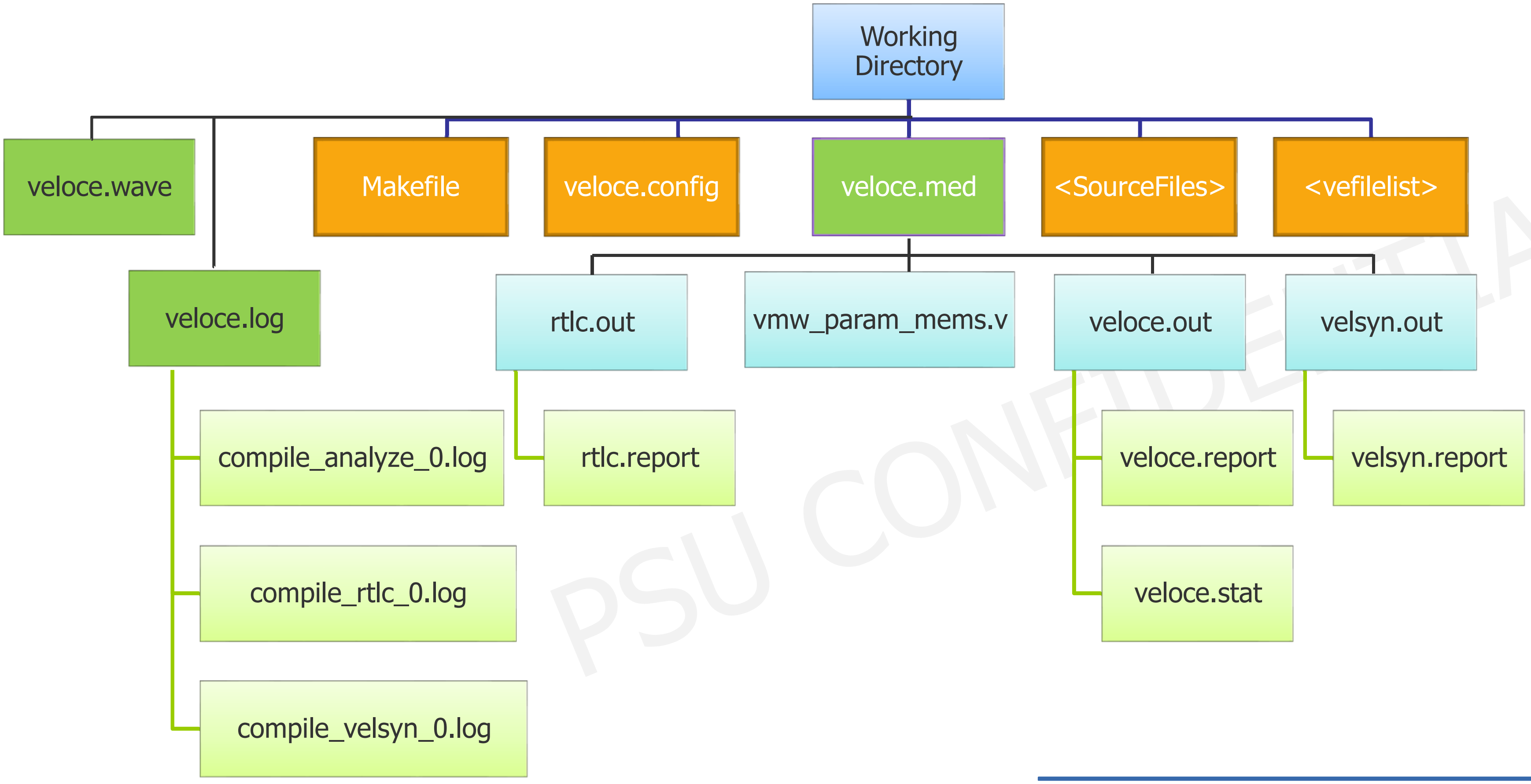
Directory Structure

- veloce.med
  - Compiled database
- veloce.log
  - Log database for both compilation and runtime
- veloce.wave
  - Waveform database for runtime

veloce.log

- Files in this directory can be used to identify errors in case of failures
- Compile logs
  - velcomp.log captures all the information presented on the screen during compilation
  - Separate detailed logs for compile phases like velanalyze/analyze, rtlc, velsyn, velcc, velgs, ssrmodelgen, riigen, visualizer
- Runtime logs
  - *velrun.transcript*, comodel logs
  - *transcript.prevsession/velrun.transcript.prevsession*
    - History of previous runtime emulation session
- Other logs
  - *compile_hvl_0.log*, *xrtl_report.txt*, *assert.log*, *cover.log*
  - *Static_Clock_Report.log*

Directory Structure



VELOCE USE MODES

Veloce Use Modes

Veloce use modes can be broadly divided into two categories:

1. **TBX Mode**: TestBench Xpress /Transaction Based Acceleration

2. **ICE Mode**: In-Circuit Emulation

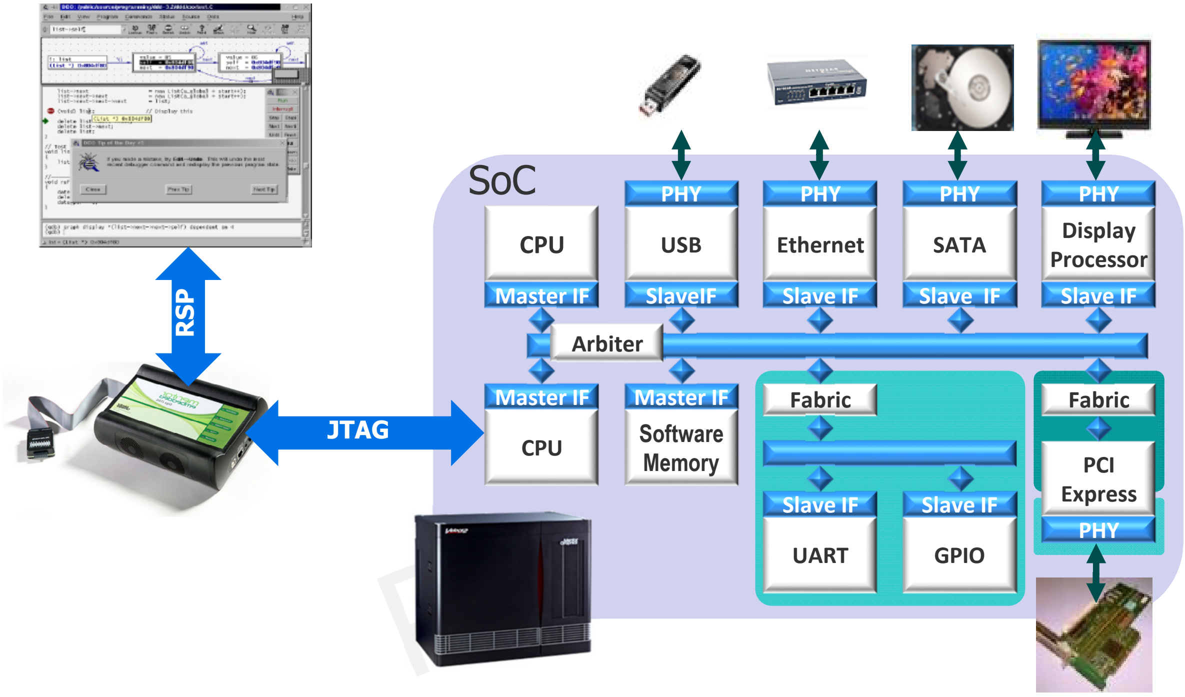
ICE/STANDALONE MODE

ICE Mode

1. Veloce provides IO ports and cables which connect the DUT to external physical devices allowing real-world stimulus during emulation.
2. You can compile HDL testbenches with your DUT and run the emulation with no external interface or testbench (aka **Target-Less mode**)

Traditional In Circuit Emulation

Embedded SW Debugger



RSP

JTAG

SoC

PHY	PHY	PHY	PHY	PHY
CPU	USB	Ethernet	SATA	Display Processor
Master IF	SlaveIF	Slave IF	Slave IF	Slave IF

Arbiter

Master IF — CPU

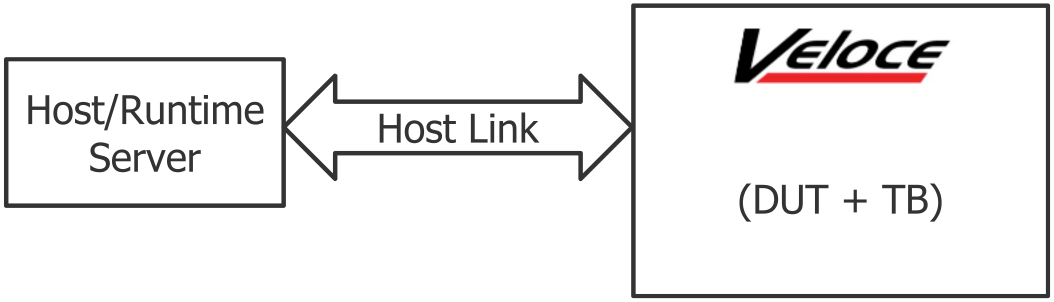
Master IF — Software Memory

Fabric — Slave IF — UART; Slave IF — GPIO

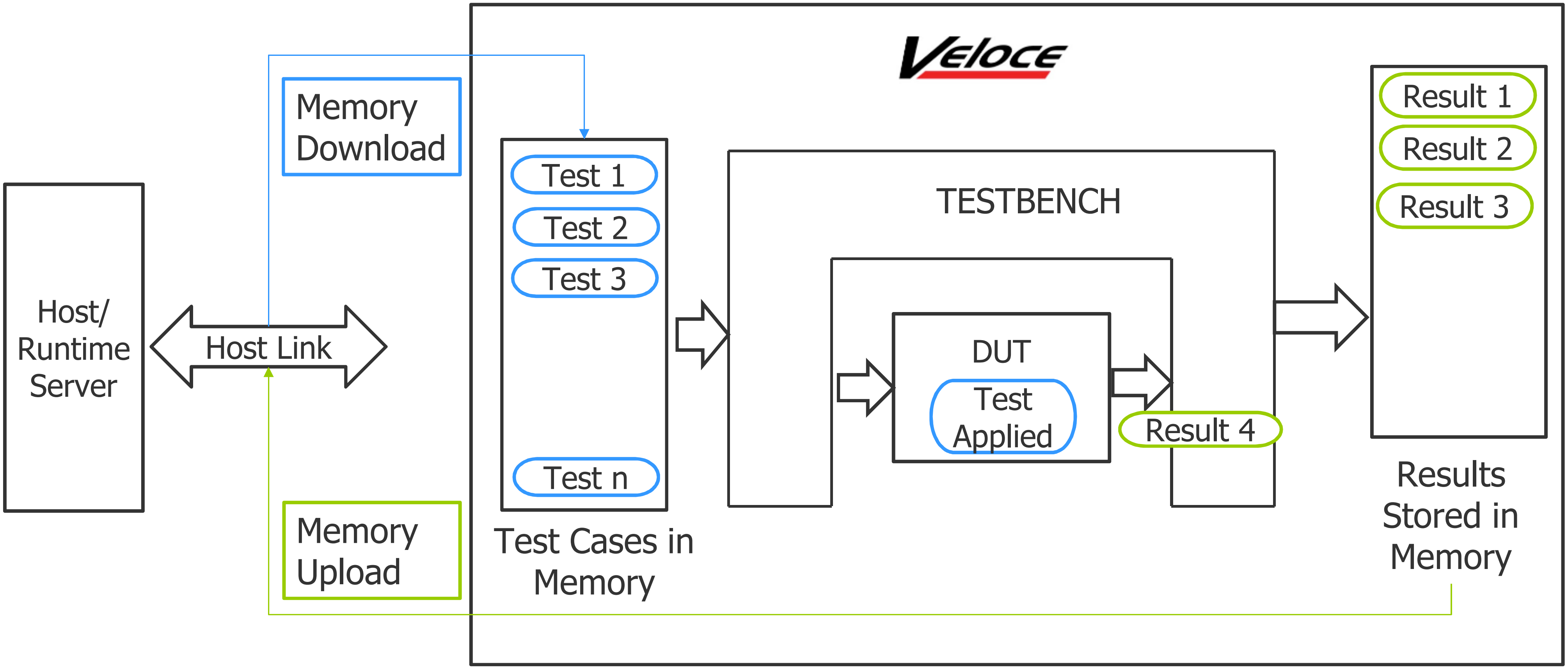
Fabric — PCI Express — PHY

Standalone Mode – Legacy (Targetless)

- No Comodel link
- No external devices
- DUT and Testbench are synthesized
- Everything runs on the Veloce emulator
- Test data is downloaded to memory before emulation
- Trace and memory data is uploaded to host after emulation



Legacy (Targetless) Example



Restricted © 2017 Mentor Graphics Corporation

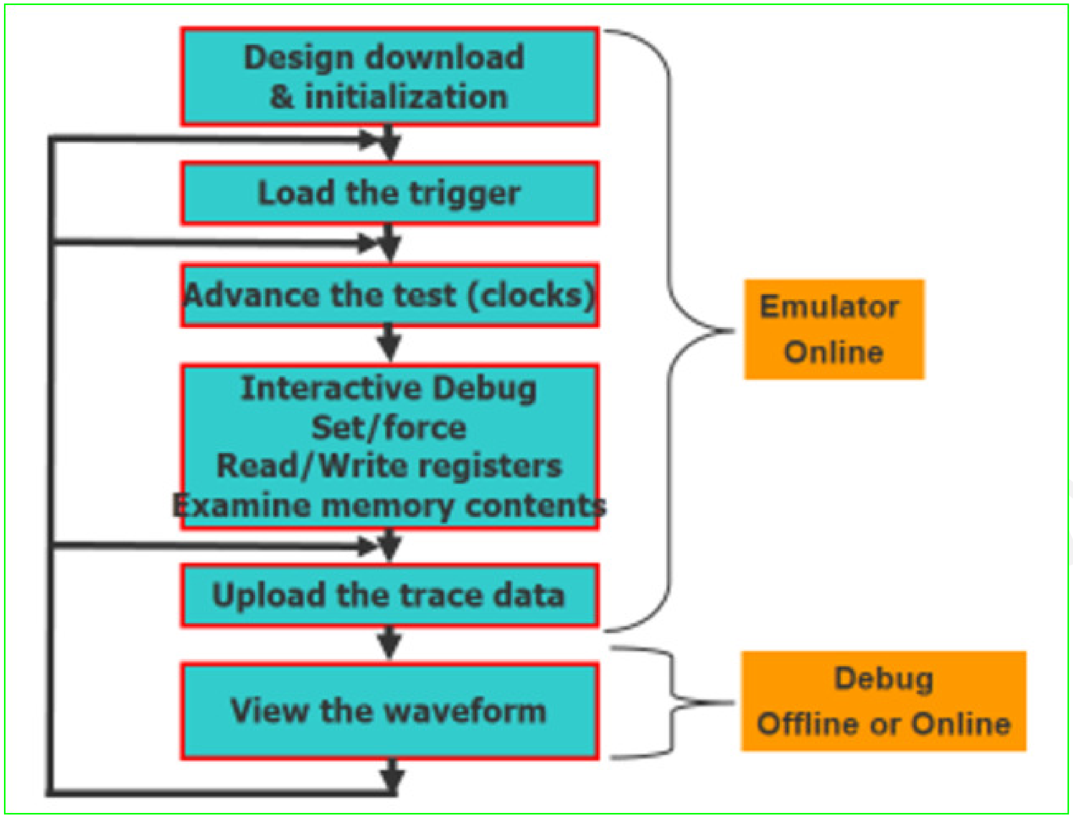
ICE Mode – Merits and Demerits

- Merits
  - Provides the most possible speedup
  - Connection to external target device

- Demerits
  - No interactive control on test stimulus
  - No on-the-fly checking and scoreboarding
  - Writing synthesizable testbench is not always possible

RUNTIME FLOW

Runtime Flow

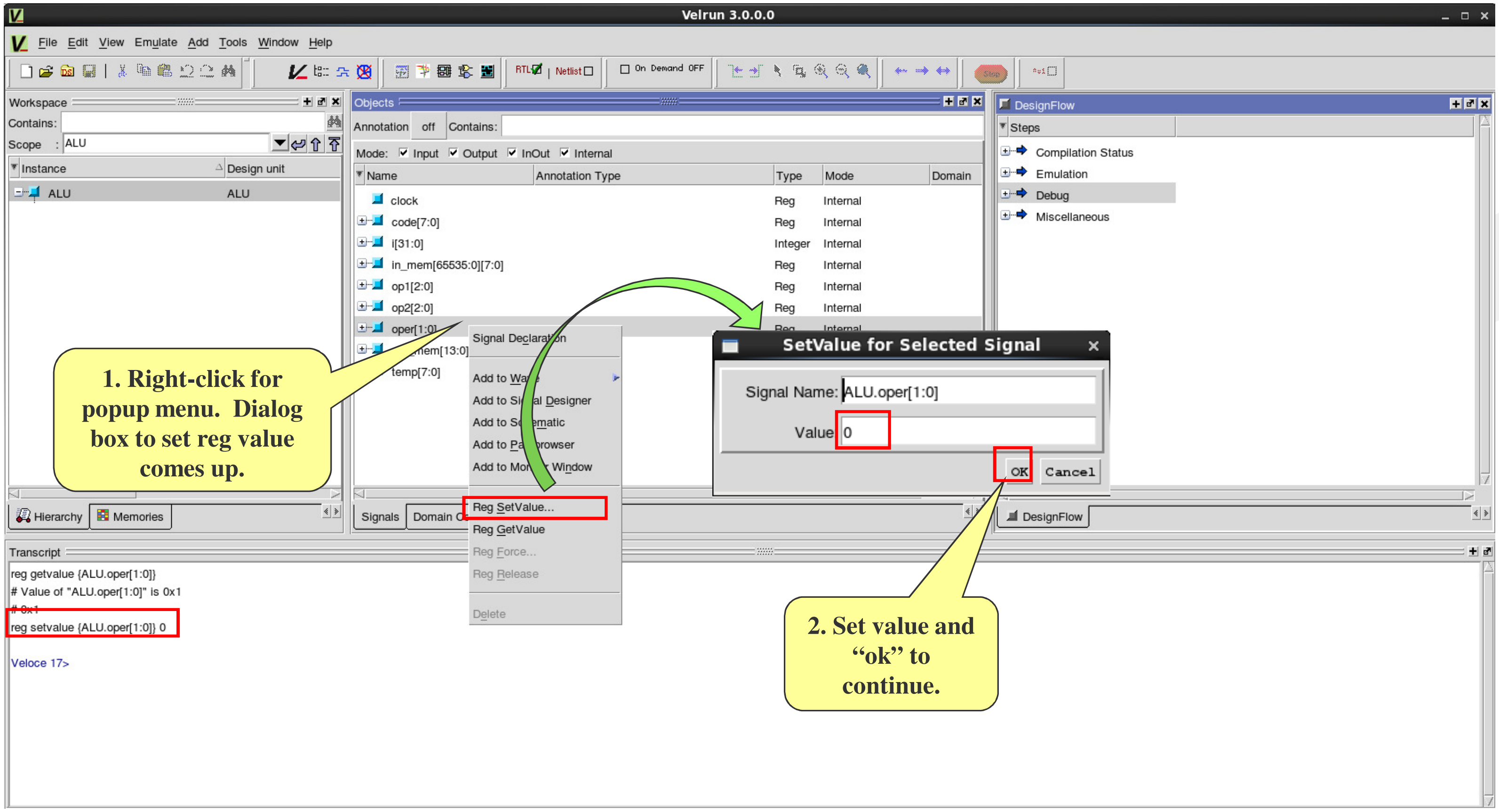


velview

- Can perform GUI operations
  - Waveform view, schematic, assertion tab
  - velview –tracedir can be used on trace directories for post-process debug
- velview –attach mode can connect to a running session

- TCL Commands
  - reg setvalue {top/dataIn[15:0]} 0
  - reg getvalue {top/dataIn[15:0]}

Emulation – Set Register Value



Name	Type	Mode
clock	Reg	Internal
code[7:0]	Reg	Internal
i[31:0]	Integer	Internal
in_mem[65535:0][7:0]	Reg	Internal
op1[2:0]	Reg	Internal
op2[2:0]	Reg	Internal
oper[1:0]	Reg	Internal

Signal Name: ALU.oper[1:0]

Value: 0

```
reg getvalue {ALU.oper[1:0]}
# Value of "ALU.oper[1:0]" is 0x1
# 0x1
reg setvalue {ALU.oper[1:0]} 0

Veloce 17>
```

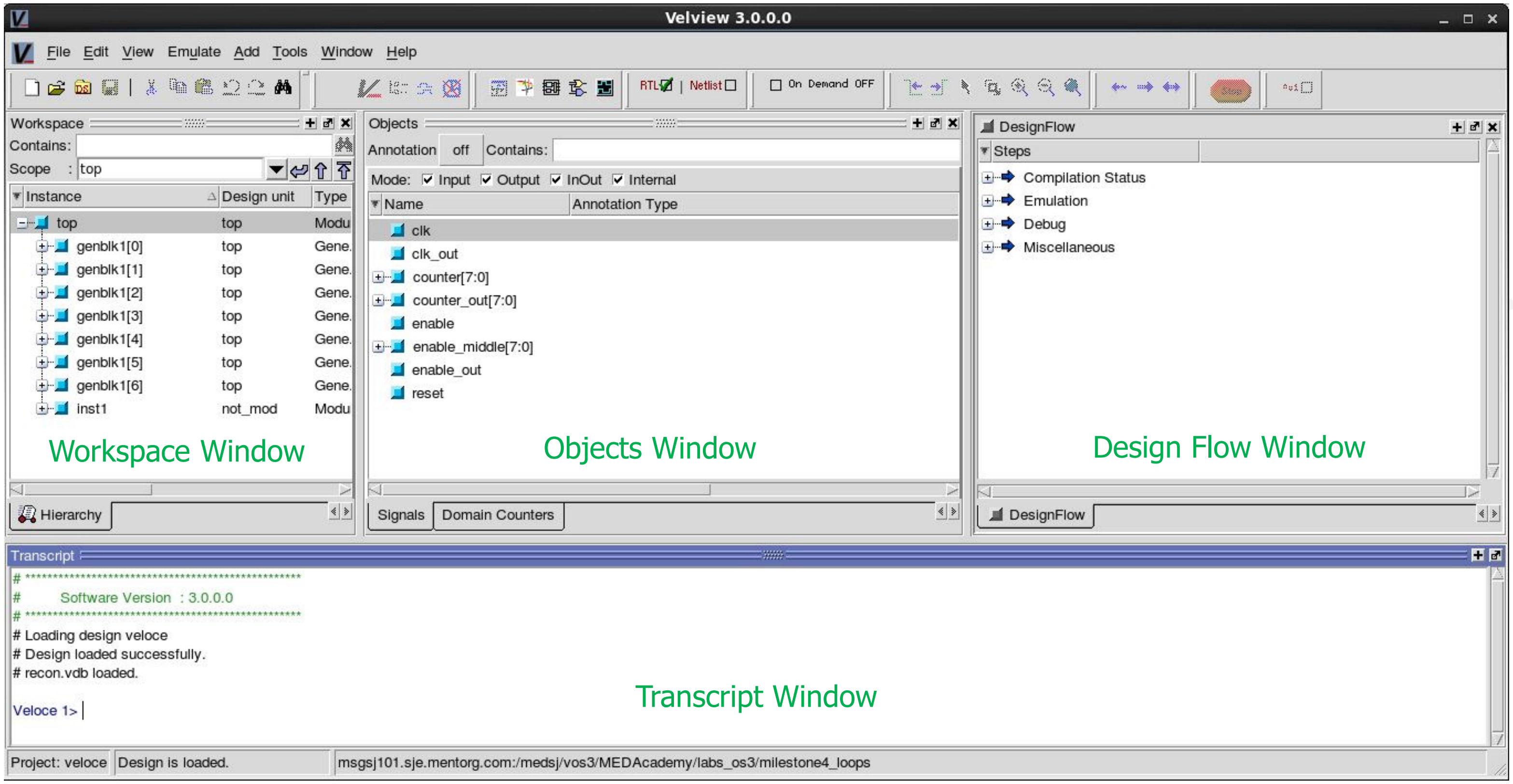
1. Right-click for popup menu. Dialog box to set reg value comes up.

2. Set value and “ok” to continue.

Restricted © 2017 Mentor Graphics Corporation

VISIBILITY

Velview



Instance	Design unit	Type
top	top	Modu
genblk1[0]	top	Gene.
genblk1[1]	top	Gene.
genblk1[2]	top	Gene.
genblk1[3]	top	Gene.
genblk1[4]	top	Gene.
genblk1[5]	top	Gene.
genblk1[6]	top	Gene.
inst1	not_mod	Modu

Name	Annotation Type
clk	
clk_out	
counter[7:0]	
counter_out[7:0]	
enable	
enable_middle[7:0]	
enable_out	
reset	

Steps

- Compilation Status
- Emulation
- Debug
- Miscellaneous

```
# ****************************************************
#        Software Version  : 3.0.0.0
# ****************************************************
# Loading design veloce
# Design loaded successfully.
# recon.vdb loaded.

Veloce 1>
```

Project: veloce | Design is loaded. | msgsj101.sje.mentorg.com:/medsj/vos3/MEDAcademy/labs_os3/milestone4_loops

References

- Veloce User Guide, Software Version 3.0.1, May 2015
- Veloce Languages and Communication Channels User Guide, Release Version 3.0.1, May 2015
- More Examples in $VMW_HOME/examples
- Documents in $VMW_HOME/doc/pdfdocs

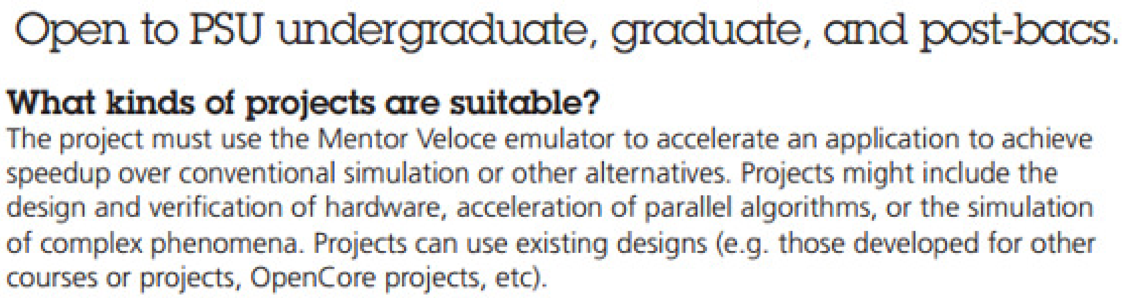
Important!

- PSU Specific examples and documents in $PSU_EXAMPLES

  Recommended documents to read (in order)
  1. Veloce_SetupUsage_OS3.pdf
  2. Veloce_Standalone_Flow.pdf
  3. Veloce_TBX_Mode_Intro.pdf
  4. Veloce_TBX_Mode_Flow.pdf
- www.verificationacademy.com
  - Advanced verification topics UVM, Coverage. Need to sign up with pdx.edu email
- Secure VNC Technical Support – The CAT

NEED FOR SPEED EMULATION CONTEST

Need for Speed Emulation Contest!



Open to PSU undergraduate, graduate, and post-bacs.

What kinds of projects are suitable?

The project must use the Mentor Veloce emulator to accelerate an application to achieve speedup over conventional simulation or other alternatives. Projects might include the design and verification of hardware, acceleration of parallel algorithms, or the simulation of complex phenomena. Projects can use existing designs (e.g. those developed for other courses or projects, OpenCore projects, etc).

Mentor®
A Siemens Business

Maseeh College of Engineering and Computer Science
PORTLAND STATE UNIVERSITY

Mentor®

A Siemens Business

www.mentor.com