Emulation for Functional Verification

Introduction

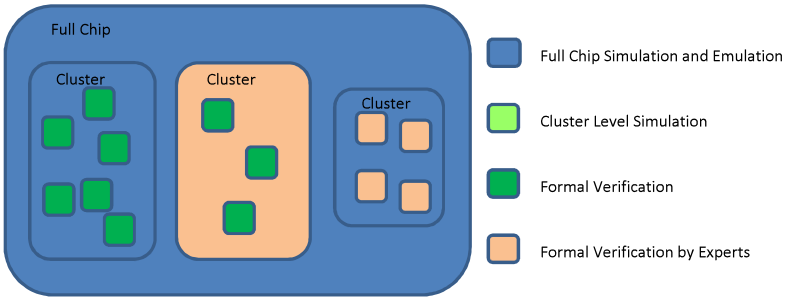
Simulation: Interpretation of HDL design by software program
Emulation: Execution of synthesized HDL design by hardware

Introduction to Emulation Outline

- **Preliminaries**
  - Functional Verification using Emulation
  - FPGAs
  - Concurrency: Begin/End vs Fork/Join and more
- **The Mentor Graphics Veloce Emulator Hardware**
  - Getting connected
  - Documentation
    - Tutorials
    - Videos
    - Manuals
  - Stand-Alone Mode
  - TBX Mode
- **Your assignments**
  - Tutorials
  - Homework
  - Class project opportunities
- **Future (post class) exploration**
  - Need For Speed Competition!
  - 1 credit research/project possibilities

Functional Verification using Emulation

- Exhaustive testing using simulation not possible
  - Even 32 bit multiply requires > 100 years to check
- PreSilicon Verification must thoughtfully select achievable objectives
  - Actively monitor progress of constraints/checking/coverage
  - Regularly "restart" or run regressions for design changes
  - Rely on huge, expensive compute server farms to run many random tests in parallel
  - Apply formal methods approaches where possible
  - Utilize Emulation!
    - Potentially orders of magnitudes faster than simulation

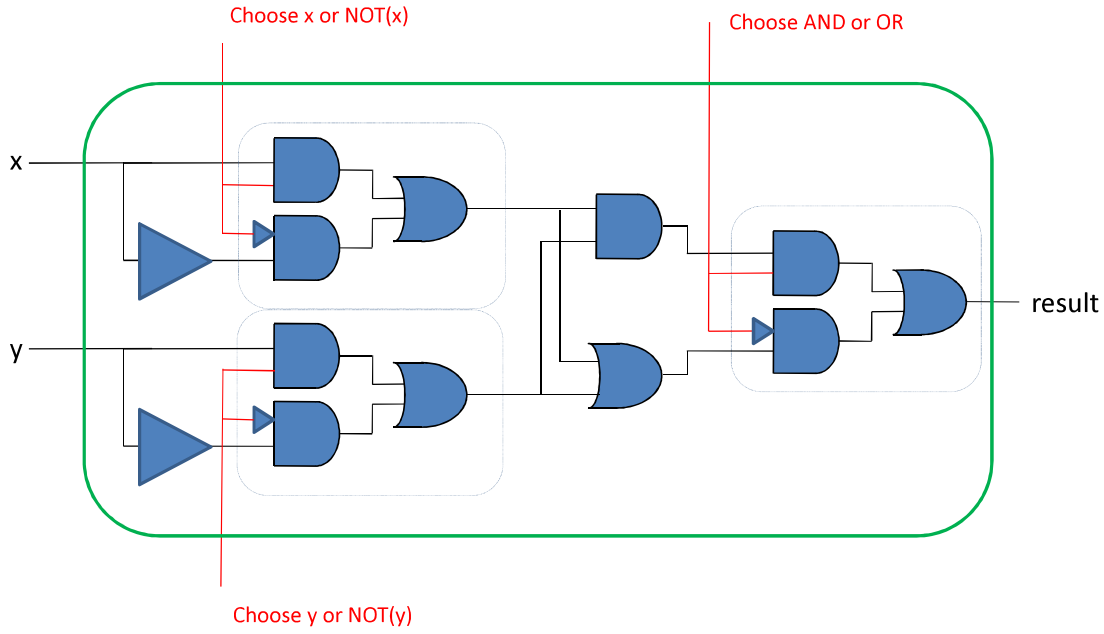


Evolving Pre-Silicon Functional Verification

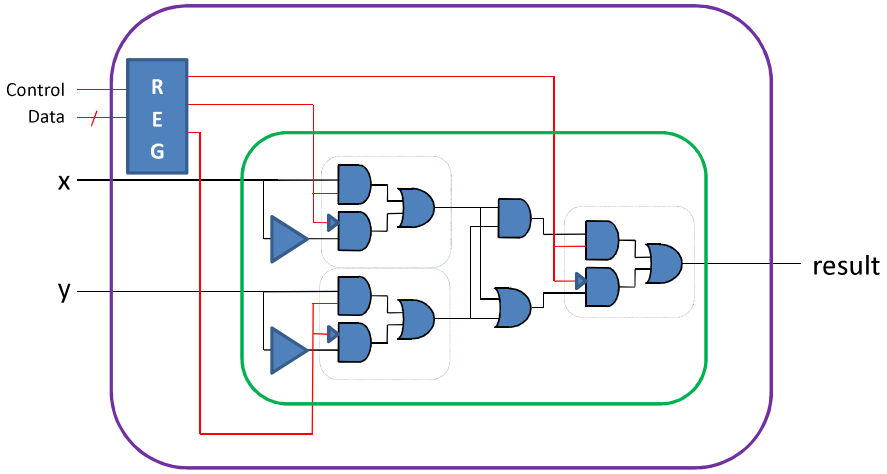
Emulation Hardware

- Emulation Hardware: specialized w/massive array of reprogrammable gates
  - Field Programmable Gate Array (FPGA)
    - Large array of gates that can be programmed/configured to create an physical representation of a circuit design implementation.
  - Typically Emulation systems also include built in features such as memories, controllers, communication interfaces, etc
- Design is specified in HDL and synthesized to RTL netlists (gate-level logic)
  - HDL either Verilog or System Verilog for the Mentor Graphics Veloce
  - FPGA synthesis produces bits to control gates and fill registers/memory
- New considerations
  - Designs must be synthesizable
    - Many advanced HDL programming constructs are not supported
    - Validation Test bench typically defined behaviorally
  - Design+TB must fit within space emulator provides
  - Visibility of design internals adds costs
    - Longer traces require more disk space to store
  - Costs of not developing w/emulator in mind

Abstracted View of FPGA



Abstracted View of FPGA



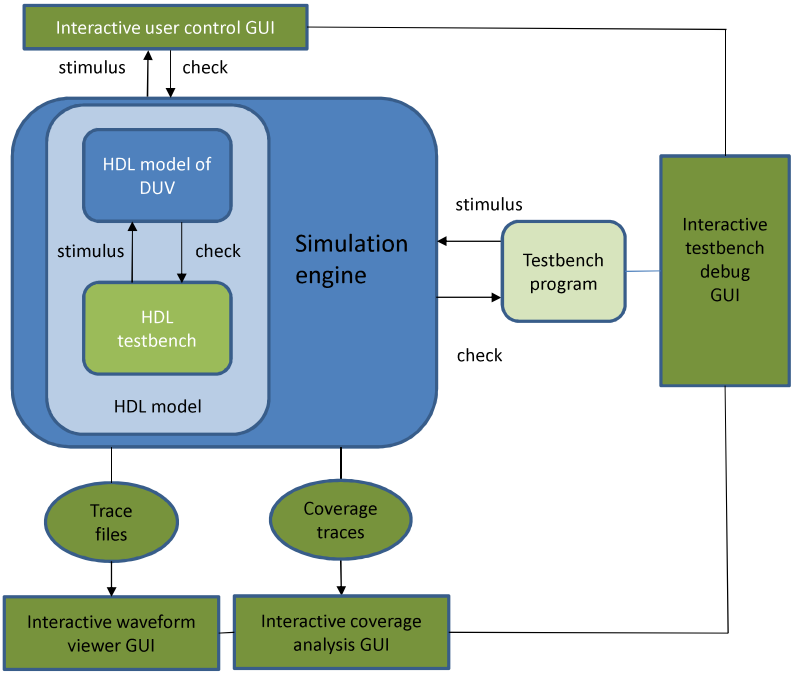
Two step process:

1) Write to Register to setup desired function
2) Drive inputs

Verilog

- Supports both structural and behavioral code
  - Structural: circuit components
    - logic gates
  - Behavioral : programming statements
    - loops, If-then-else, test generation code
    - No corresponding circuit component
    - Occur in initial and always blocks
  - Procedural assignment may be blocking (=) or nonblocking (<=)
    - blocking: statement must be executed before next statement
    - nonblocking: RHS evaluations first, then assignments done
- During simulation
  - Initial and Always blocks execute in parallel
  - All blocks begin execution at time zero
  - Initial blocks execute once, Always blocks repeat

Simulation tool overview



EMULATORS

- Mentor Graphics: Veloce

- Synopsys/EVE: Zebu

- Cadence: Palladium

From presentation by Rahul Wagh (former PSU grad student)

EMULATORS: COMPARISON

Vendor	Cadence, Mentor Graphics	Synopsys/EVE	Aldec, Bluespec
Architecture	Processor-based	FPGA-based	FPGA-based
Price/Gate	2 – 5 cents	0.5 – 2 cents	0.25 – 1 cents
Dedicated Support	Yes	Mixed	No
Speed Range (cycles/sec)	100K – 2M	500K – 5M	500K – 20M
Compile time	10M –30M gates/hr	25M–100M gates/hr	1M–15M gates/hr
Partitioning	Automated	Automated	Semi-automated
Visibility	Full visibility	Static/dynamic	Static/dynamic
Verification language native support	C++, SystemC, Specman e, SystemVerilog, OVM, SVA, PSL, OVL	Synthesizable Verilog, VHDL, SystemVerilog	Synthesizable Verilog, VHDL, SystemVerilog
Memory	Up to 1TB	Up to 200GB	Up to 32GB
Users	1 – 512 users	1 – 49 users	1 user

From presentation by Rahul Wagh (former PSU grad student)

VELOCE2 EMULATORS: SPECS

Specification	Description
Capacity	Up to 2 billion gates
Number of users	Up to 128
Design Languages	Verilog, VHDL, SystemVerilog, gate-level netlist, and encrypted IP
Testbench Languages	C, C++, SystemC, SystemVerilog
Methodologies	UVM, OVM, TLM, VMM
Assertion Languages	SVA, PSL, OVL, QVL
Fast Compile	Up to 40 MG per hour
Asynchronous clocks	Can be supplied using functional generators or external PLLs
OS support	RH 4 & 5, SUSE 10 & 11
Power	About 11KW for a fully loaded system

From presentation by Rahul Wagh (former PSU grad student)

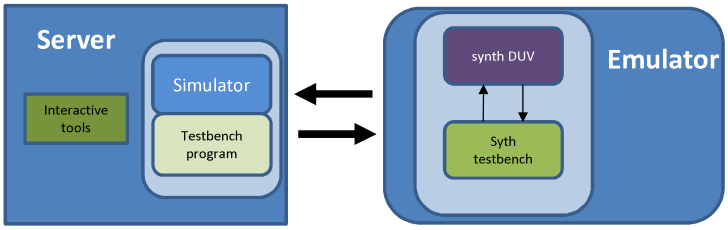
(Veloce) Emulation

Virtual Silicon Environment

- Goal to verify designs under "real world" situations
  - Early system integration
  - Apply live stimulus or connect real devices
  - Enable software development and debug
  - SW-HW co-verification, boot the OS, run deep sequences
  - Async clock modeling to verify interactions
  - Performance optimizations and stress testing
  - Run compliancy suites

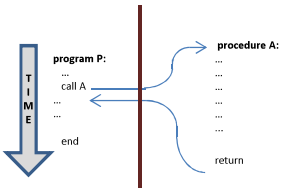
- Bug reproduction / root cause analysis
  - 100% visibility of all then nodes in the design
  - Simulation like interactive debug and intuitive GUI
  - Allow error injection
  - Faster turnaround

Server (velocesolo) and Emulator (velocesolo1)

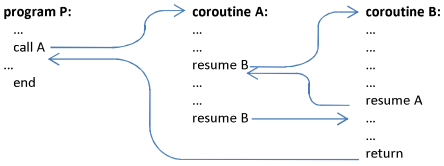


Concurrent Execution Models

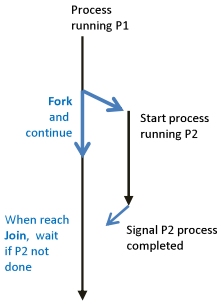
- Simple procedure call



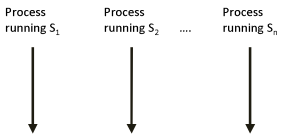
- Co-routines



- Fork and Join

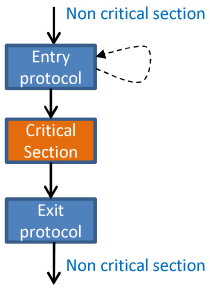


- Cobegin

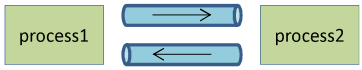


Concurrency: information sharing

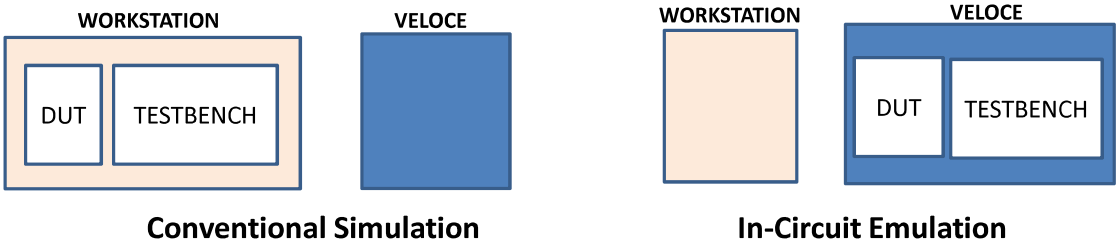
- Process Synchronization
  - Shared Memory
    - Busy-waiting
    - Semaphore
    - CCR
    - Monitors

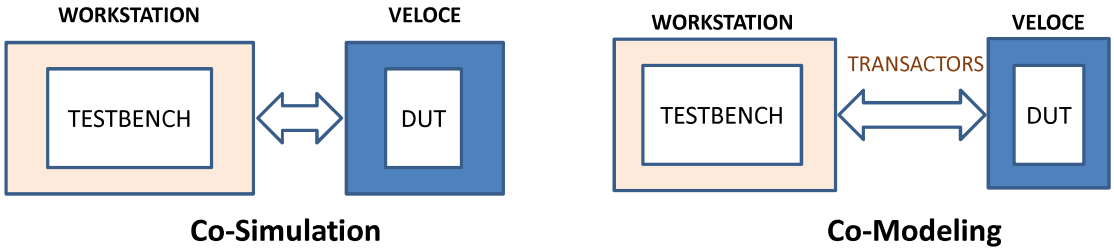


  - Message Passing
    - Communication channels consisting of sender and receiver
      - Direct naming, mailboxes (ports)
    - Sending and receiving may be blocking actions or non blocking
      - Buffered/asynchronous
      - Nonbuffered/synchronous
    - Higher level constructs
      - Remote Procedure calls
      - Atomic Transactions



Veloce Modes

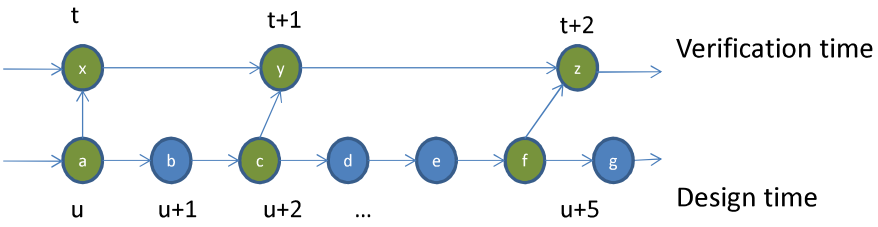




From presentation by Rahul Wagh (former PSU grad student)

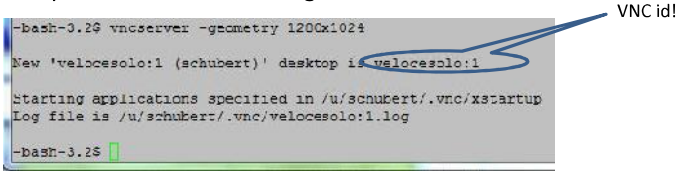
Interaction Between HVL and HDL

- Considerations
  - Communication
  - Speed differential
    - Where is the code running?
  - Visibility differential
    - What trace information do you have? How do you debug?
  - Virtual time



Connecting to Veloce Server

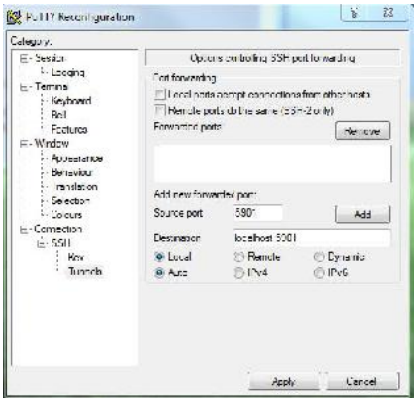
- Using PuTTY
  - set host to velocesolo.ece.pdx.edu and connect/login
  - Start a VNC session



VNC id!

  - First time: edit ~/.vnc/xstartup and uncomment two lines
  - Exit PuTTY

- Start VNC viewer and connect using full path to vncserver id
  - For example, using TightVNC: velocesolo.ece.pdx.edu:1
  - If you get an authentication error, run PuTTY to open secure connection
    - Set Connection/SSH/Tunnels parameters
      Source port to 59XX (5900 + your VNCid)
      Destination to localhost:59XX
    - Press Add button
    - Go back to Session,
      enter host *velocesolo.ece.pdx.edu*
      press Open
    - Start VNC server again



Connecting to Veloce Server

- Only one person at a time can use the Veloce Emulator.
  - Add to .bashrc

    alias v='whoison –emul velocesolo1'

- Veloce Server disk shared with other ECE machines:
  - On ubuntu: /stash/veloce/username
    on Windows \\stash\veloce\username

Documentation: https://veloce.ece.pdx.edu/

Introduction

This web site is primarily for background information on Veloce, including tutorials, demos, and manuals.

A companion Wiki web site serves as a project repository and has discussion groups on a variety of Veloce topics. If you are having problems with Mentor Veloce, check the discussion forums at that site. You'll need to login (use the same login/password as you did for this site) to see the discussions. The site also hosts student projects, including SVN repositories for source files.

Tutorials

The following tutorial will acquaint you with accounts, directories, and tools you need to use the Mentor Veloce system at PSU. Note that HDL Link is no longer supported by Mentor, so these tutorials and training materials have been removed.

Using Mentor Veloce at PSU

Latest (2012) Mentor Veloce training slides

Use model suites (Wishbone)

Standalone Mode Tutorial Using Verilog Adder Example

Tutorial

Adder Verilog Source

Adder Testbench Verilog Source

Training & Labs

To untar the lab files use the Unix command "gtar -xvzf filename". Some of these files are quite large. Be judicious in your use of disk space.

Basic Veloce

Veloce Training Slides

Labs

Documentation: https://veloce.ece.pdx.edu/

TBX

- TBX Training Slides
- Labs
- The following are videos of the TBX training held on May 27, 2010. Note access to videos requires an ODIN account
- Part 1 in Rich Media format
- Part 2 in Rich Media format
- Part 1 in Vodcast format
- Part 2 in Vodcast format
- Part 1 in Enhanced Podcast format
- Part 2 in Enhanced Podcast format
- Part 1 in Podcast format
- Part 2 in Podcast format

OVM

- OVM Slides

Manuals

- Veloce Family Datasheet
- Install Notes
- Release Notes
- VMP User Guide
- HDL Link
- TBX User Guide
- PCI Board
- Reference Manual
- Solo Hardware Reference Manual

SCE-MI 2.0 (Standard Co-Emulation Modeling Interface)

- **SCE-MI 2.0 Reference Manual**