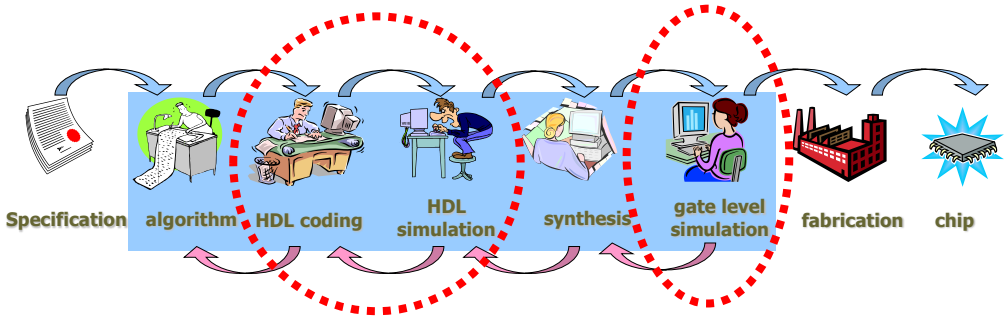
Introduction to HDL
(Hardware Description Language)

May 2014

Ando KI

Contents

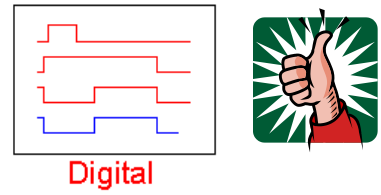
The phases HDL is used

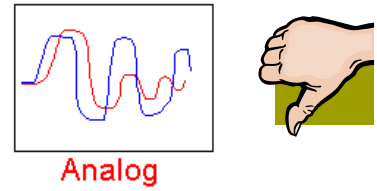


HDL: Hardware description language

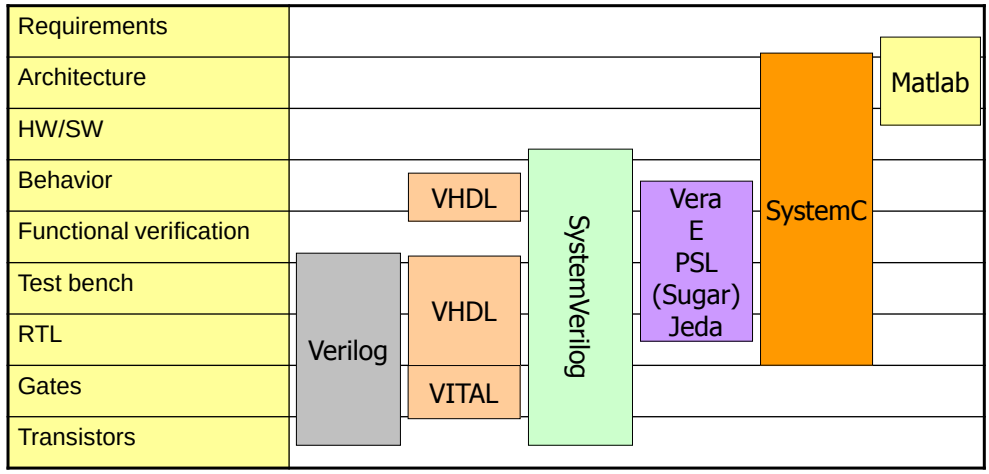
What is HDL?

- HDL is a concurrent language aimed at modeling digital hardware.





Design/verification language spectrum



Source – SystemC: From the ground up.

HDL (Hardware description language)

- Verilog
  - Verilog 1.0: IEEE Std. 1364-1995 – i.e., Verilog-1995
  - Verilog 2.0: IEEE Std. 1364-2001 – i.e., Verilog-2001
  - Verilog 3.0: IEEE Std. 1364-2005 – i.e., Verilog-2005
- VHDL
  - IEEE Std. 1076-1987
  - IEEE Std. 1076-1993 – i.e., VHDL '93
  - IEEE Std. 1076-2000
  - IEEE Std. 1076-2002
- SystemVerilog
  - IEEE Std. 1800-2005
- SystemC
  - 1999-09-27 Open SystemC Initiative announced
  - 2000-03-01 SystemC V0.91 released
  - 2000-03-28 SystemC V1.0 released
  - 2001-02-01 SystemC V2.0 specification and V1.2 Beta source code released
  - 2003-06-03 SystemC 2.0.1 LRM (language reference manual) released
  - 2005-06-06 SystemC 2.1 LRM and TLM 1.0 transaction-level modeling standard released
  - 2005-12-12 IEEE approves the IEEE 1666–2005 standard for SystemC
  - 2007-04-13 SystemC v2.2 released
  - 2008-06-09 TLM 2.0 released

History of Verilog HDL

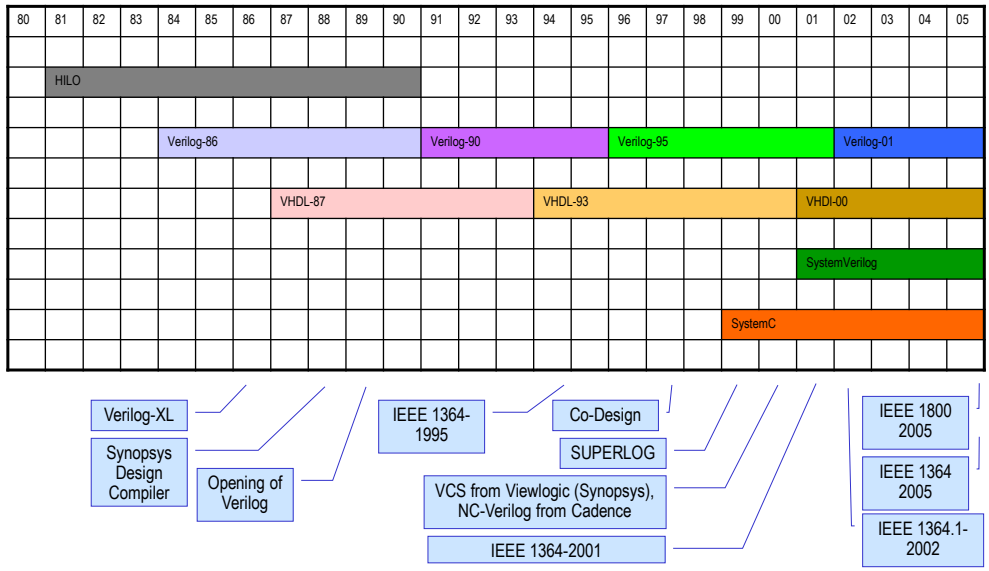
- 1983
  - Verilog HDL and Simulator released by Gateway Design Automation, which was founded at 1981.
- 1985
  - Verilog-XL simulator released
- 1987
  - Proprietary Verilog behavioral language used by Synopsys for its synthesizer
- 1988
  - Synopsys releases Verilog RTL synthesizer, Design Compiler
- 1989
  - Gateway acquired by Cadence Design Systems
- 1989
  - Verilog is donated to OVI (Open Verilog International, which was merged into Accellera with VHDL international, 2000).
- 1990
  - Verilog and PLI (Programming Language Interface) donated to OVI, which was an industrial consortium.
  - Verilog-XL as a separate product
- 1993
  - Verilog submitted to (reviewed and adopted by) the IEEE as IEEE Std. 1364. →It becomes Verilog Std.1364-1995.
  - 85% of design submitted to ASIC foundries was Verilog.
- 1995
  - IEEE Std. 1364
- 2001
  - IEEE Std. 1364-2001
- 2002
  - IEEE Std. 1364.1
    - Verilog Register Transfer Level Synthesis
- 2005
  - IEEE Std. 1364-2005

History of VHDL

- 1980
  - VHSIC program launched
  - Objective was to achieve significant gains in VLSI technology by shortening the time from concept to implementation (18 months to 6 months)
- 1981
  - Initiated by US DoD to address hardware life-cycle crisis
- 1983-85
  - Development of baseline language by Intermetrics, IBM and TI
- 1985
  - VHDL Revision 7.2
- 1986
  - All rights transferred to IEEE
- 1987
  - Publication of IEEE Std 1076-1987
  - The DoD mandated that all digital electronic circuits be described in VHDL.
  - The F-22 advanced tactical fighter aircraft was one of the first major government programs to mandate the use of VHDL descriptions for all electronics subsystems in the project.
- 1987
  - Mil Std 454 requires comprehensive VHDL descriptions to be delivered with ASICs
- 1993
  - Revised standard (named VHDL 1076-1993)

VHDL: Very high speed integrated circuit Hardware Description Language

History of HDL

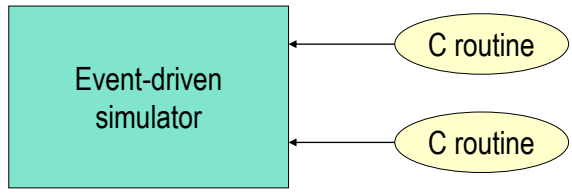


Other HDL

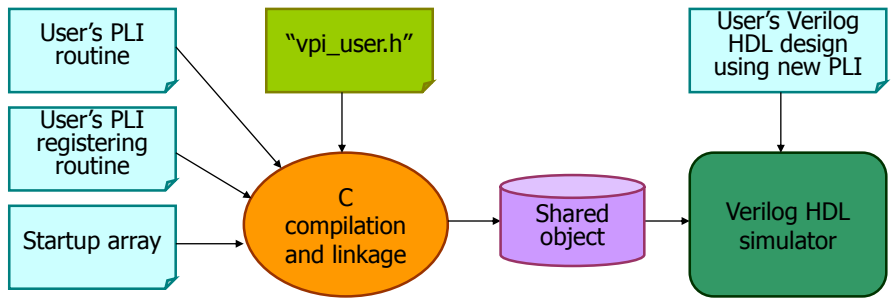
- ABEL (Advanced Boolean Equation Language)
  - Simplified HDL for PLD, such as Xilinx FPGA
- AHDL (Altera Hardware Description Language)

Language extensions

- HDL provides interfacing mechanism to high-level languages.
  - PLI (Programming Language Interface) for Verilog.
  - VPI (Verilog Procedural Interface)
  - FLI (Foreign Language Interface) for VHDL.
  - VHPI (VHDL Procedural Interface)
  - DPI (Direct Programming Interface) for SystemVerilog



How it works at a glance: VPI case



History of Verilog PLI

- 1985: TF routines
  - Verilog PLI as Verilog-XL simulator
  - A procedural interface to Verilog simulator through system tasks and system functions.
  - TF stands for Task and Function.
  - “veriuser.h”
- 1988: ACC routines
  - A procedural interface to access information within in a simulation.
  - ACC stands for access.
  - “acc_user.h”
- 1990: OVI PLI 1.0
  - TF and ACC libraries
- 1993: OVI PLI 2.0
  - OVI intended that PLI 2.0 would completely replace the PLI 1.0.
- 1995: IEEE 1364-1995
  - TF and ACC routines from OVI PLI 1.0.
  - VPI (Verilog Procedural Interface) routines from OVI PLI 2.0 are a super set of TF and ACC routines.
  - “vpi_user.h”
- 2001: IEEE 1364-2001
  - VPI routines enhanced to support new Verilog-2001 features.
  - TF and ACC routines updated, but no new TF/ACC routines added.

Enhanced API of SystemVerilog

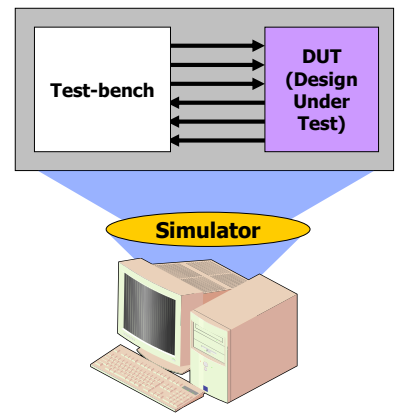
- Enhanced API of SystemVerilog 3.1
  - Direct Programming Interface (DPI) ← DirectC interface from Synopsys
  - Assertions VPI: VPI extensions for assertions ← Assertions from Synopsys
  - Coverage Metrics VPI: VPI extensions for coverage ← Coverage from Synopsys
  - Waveform Reader VPI
  - SystemVerilog Extensions to VPI
- Why new API?
  - VPI and PLI are not easy interfaces to use.
  - VPI and PLI are not symmetric.
    - Verilog can invoke C functions, but C function cannot invoke Verilog functions.
  - SystemVerilog has new data types.
  - SystemVerilog includes assertions.
  - Coverage driven test must be supported.
- For more information
  - www.eda.org/sv-cc: SystemVerilog C-interface Committee

Simulation and simulator

- **Verification** is a process to be sure that the system's operation is correct.
  - In the context of electronic hardware design, there are many verification methods such as simulation, emulation, formal verification, prototyping and so on. The goal of verification is to ensure that the design meets the functional requirements as defined in the functional specification.
- As a kind of verification, **simulation** is a process to create a model that responds in a similar way to the manufactured and tested design.
- Typically, **simulator** is a software utility for executing models within a computer.

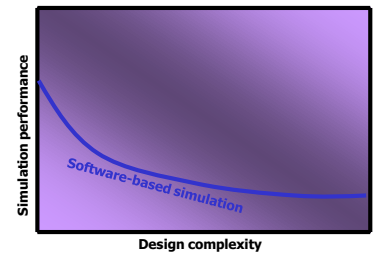
Software simulator (1/2)

- Dynamic verification method
  - Requires test-bench
  - Must be run on the computer
- Bugs are found by running the implemented design → simulation
- Thoroughness depends on the test vector used → quality of test-bench
- Some parts are tested repeatedly while other parts are not even tested.



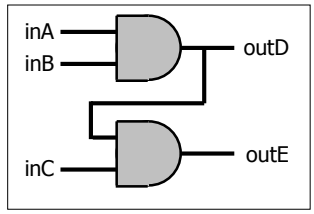
Software simulator (2/2)

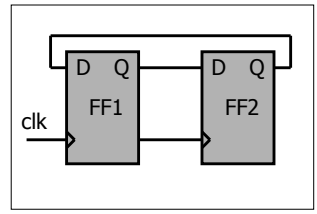
- Pros
  - The design size is limited only by the computing resource.
  - Simulation can be started as soon as the design is ready in HDL.
  - Set-up time is minimal.
- Cons
  - Slow (~100cycles/sec)
  - Speed is inverse-proportional to the size of the design
  - It is hard to know how much of the design has been tested.
    - Need coverage test.

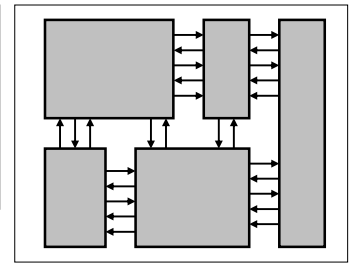


Logic simulation

- Need special language to capture parallel behavior of hardware.
- Thus, HDL (Hardware Description Language) has to be a parallel programming language, because real world objects run in parallel.
- Also, simulator is required to run a parallel program.

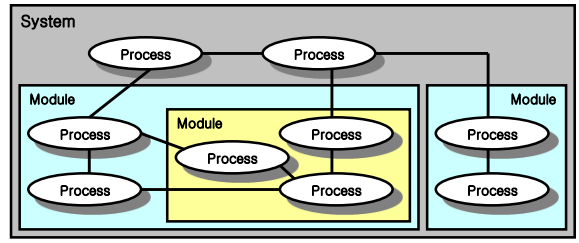






Concept of design

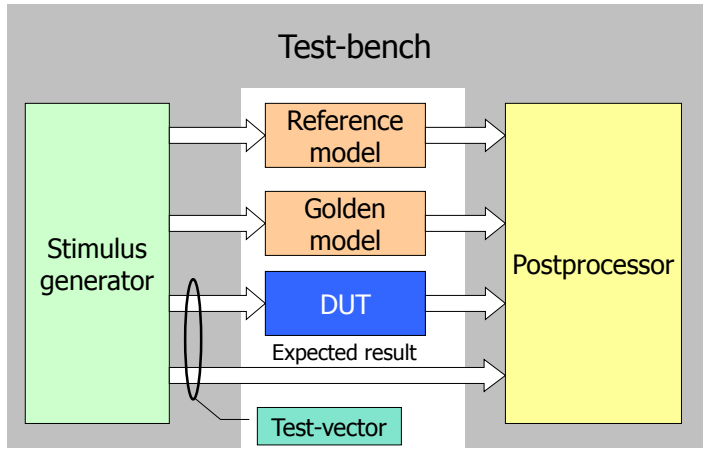
- A design consists of connected threads of execution or processes.
  - Processes are objects that can be evaluated, that may have state, and that can respond to changes on their inputs to produce outputs.
- An HDL description usually describes design behavior in terms of processes that run when triggered by input signal changes.



Test-bench (1/2)

- A test-bench is a layer of code that is created to apply input patterns (stimulus) to the DUT (design under test) and to determine whether the DUT produces the outputs expected.
- A test-vector is a set of values for all the expected input ports (stimuli) and expected values for the output ports of a module under test.
- A test-bench that is created to apply inputs, sample the outputs of the DUT, and compare the outputs with the expected (golden) results is called a self-checking test-bench.

Test-bench (2/2)



Types of simulator

- Depending on how it executes models
  - Interpreted: It is effective for small design. → Obsoletes
    - Verilog-XL
  - Compiled: It requires compile time, but runs fast.
    - NC-Verilog, VCS, ModelSim
- Depending on how to handle events
  - Event-driven
  - Cycle-based

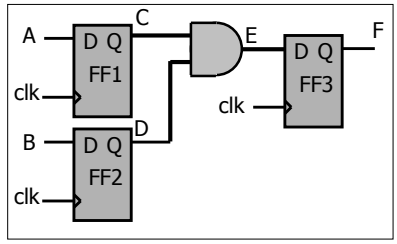
Event-driven simulation

- Take events one at a time, propagate them through a design until a steady state condition is achieved
- The design model includes functionality and timing.
- Can detect glitches in the design
- Slow than cycle-base simulator
- Types of EBS
  - Compiled-code simulator – *Cadence NC-Verilog; Synopsys Verilog Compiled Simulation (VCS); Mentor ModelSim; Synopsys VHDL System Simulator (VSS);*
  - Interpreted-code simulator – *Cadence Verilog-XL*

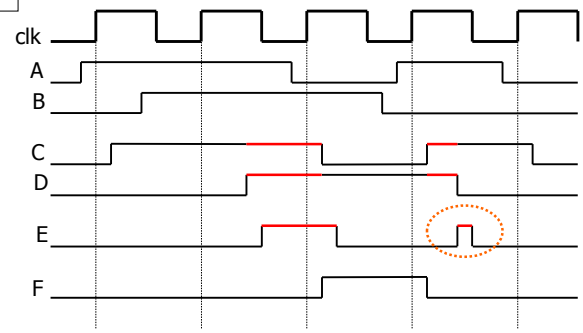
Cycle-based simulation

- No notion of time within a clock cycle
- Only function on synchronous logic
- Cannot detect glitches
- Timing verification are required (static-timing tools)
- Fast than event-simulator
- Examples
  - *Synopsys VHDL cycle-based simulator Cyclone; Quickturn/Cadence Verilog cycle-based simulator SpeedSim*

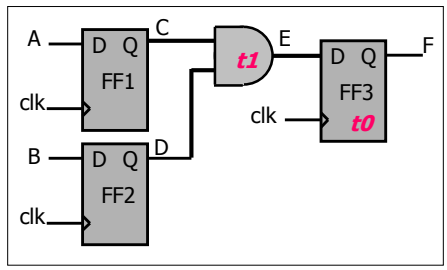
Glitch



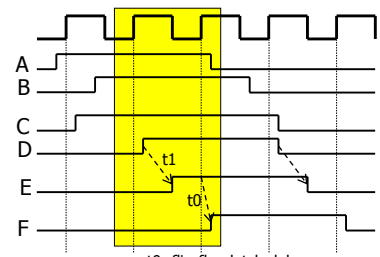
Glitch: An unwanted, spurious logic pulse of very short duration and most result from poor timing or bad combinatorial design.



EBS vs CBS

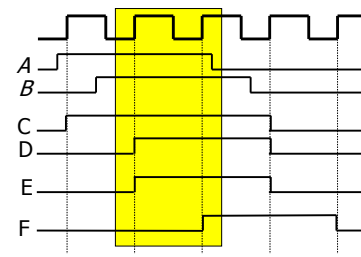


EBS



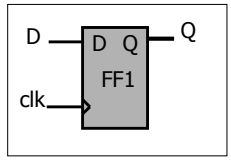
t0: flip-flop latch delay
t1: AND gate delay

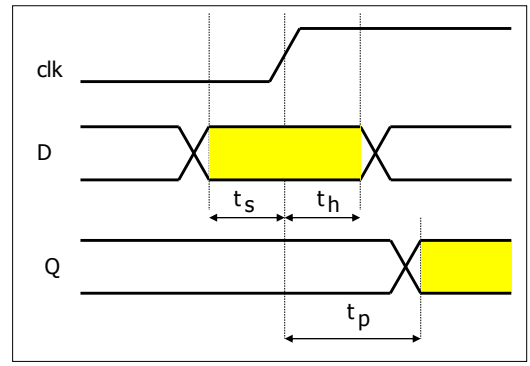
CBS



Setup and hold time violation is not detected by cycle-based simulation, i.e., static timing analysis is required.

Setup and hold time





Setup time: The time data must remain stable before clock transitions on a flip-flop.

Hold time: The length of time of a flip-flop's input data must remain stable after the each clock transition.

Violating the setup and hold time will create a metastable state, which is a condition giving a random data.

		min	typ	max
Setup		0ns		
Hold		0.2ns		
propagation	P_{LH}	2ns	6ns	10ns
	P_{HL}	2ns	6ns	10ns

Comparing EBS and CBS

	EBS	CBS
Output evaluation is done	For every input change	For every clock cycle
Computation of time delay	Performed	No
Event scheduling is done	Yes	No
Store timing information	Yes	No
Identify timing violation	Yes	No
Simulation speed	Low	10x~100x times EBS
Static timing analysis	Not required	Required
Best application	Simulation of small block and critical part of large design	Highly suitable for design regression testing; SoC with processors and peripherals
limitations		Poor performance for asynchronous design; May not handle X and Z

Source: System-on-a-chip verification – P.Rashinkar

References

- IEEE Std. 1364-2001, IEEE Standard Verilog Hardware Description Language, 2001. (http://standards.ieee.org/)
- IEEE Std. 1364.1-2002, IEEE Standard for Verilog Register Transfer Level Synthesis, 2002. (http://standards.ieee.org/)
- S. Sutherland, SystemVerilog for Design, KAP, 2005.
- S. Sutherland, The Verilog PLI Handbook, KAP, 2002.

Acronyms

- ACC: access
- ASIC: Application Specific Integrated Circuit
- API: Application Programming Interface
- BNF: Backus-Naur Form
- DIMM: Dual In-line Memory Module
- DoD: Department of Defense, USA
- DPI: Direct Programming Interface
- HDL: Hardware Description Language
- IEEE: Institute (of) Electrical (and) Electronic Engineers
- LHS: Left Hand Side
- OVI: Open Verilog International
- PCI: Peripheral Component Interconnect
- PLI: Procedural Language Interface
- RHS: Right Hand Side
- RTL: Register Transfer Level
- SDRAM: Synchronous Dynamic Random Access Memory
- TF: Tasks and Functions
- VHDL: VHSIC Hardware Description Language
- VHSIC: Very High Speed Integrated Circuit
- VITAL: VHDL Initiative Toward ASIC Libraries
- VPI: Verilog Procedural Interface