COE 405
Hardware Design Environments

Dr. Aiman H. El-Maleh

Computer Engineering Department

King Fahd University of Petroleum & Minerals

Outline

- Welcome to COE 405
- Digital System Design
- Design Domains and Levels of Abstractions
- Synthesis Process
- Objectives of VHDL
- Styles in VHDL
- Design Flow in VHDL
- Simulation Process

Welcome to COE 405

- **Catalog Description**
  - Design methodology. Hardware modeling basics. Modeling concurrency and timing aspects. Behavioral, structural, and data flow level modeling using hardware description languages (HDLs). System level modeling and design of practical processors, controllers, arithmetic units, etc. Translation of instruction sets to hardware models for software emulation. Case studies.
- ***Prerequisite:* COE 308 or consent of instructor**
- **Instructor Dr. Aiman H. El-Maleh. Room: 22/318 Phone: 2811 Email: aimane@ccse.kfupm.edu.sa**
- **Office Hours SMW 10:00-10:50, and by appointment**

1-3

Course Objectives

- **After successfully completing the course, students will be able to:**
  - Master the hardware description language, VHDL, for the design (specification, modeling, simulation, and synthesis) of digital systems using programmable logic or VLSI components.
  - Design complete digital systems starting from the concept, advancing through the modeling, simulation, synthesis, and test, by using different styles in VHDL, namely structural, dataflow, and behavioral, for describing the architecture.

1-4

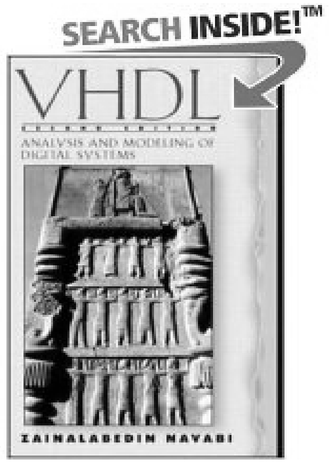
Course Learning Outcomes

- Ability to design a digital system based on VHDL including modeling, simulation, and synthesis.
- Ability to use CAD tools for the analysis and verification of digital designs.
- Ability to demonstrate self-learning capability.
- Ability to work in a team.

1-5

Text Book

- Zainalabedin Navabi, "VHDL: Analysis and Modeling of Digital Systems", McGraw-Hill, Inc., 2nd edition, 1998



1-6

Grading Policy

Assignments	15%
Quizzes	10%
Exam I	15% (Th., Mar. 29, 1:00 PM)
Exam II	20% (Th. , May 10, 1:00 PM)
Project	20%
Final	20%

- Attendance will be taken regularly.
- Excuses for officially authorized absences must be presented no later than one week following resumption of class attendance.
- Late assignments will be accepted (upto 3 days) but you will be penalized 10% per each late day.
- A student caught cheating in any of the assignments will get 0 out of 15%.
- No makeup will be made for missing Quizzes or Exams.

1-7

Course Content

- **Structured Design Methodologies: Digital System Design, Abstraction hierarchy, Types of Behavioral Descriptions, Digital Design Space & Design Decomposition.**
- **VHDL Quick Overview: Design Partitioning & Top-Down Design, Design Entities, Signals vs. Variables, Architectural Bodies, Different design views, Behavioral model, Dataflow model, Structural model.**
- **VHDL Language Basics: Lexical Elements, Data Types (Scalars & Composites), Type Conversion, Attributes, Classes of objects. Operators & Precedence, Overloading.**

1-8

Course Content

- Signals, Delays & Concurrency: Variables vs. Signals, Sequential vs. Concurrent Constructs, Signal Propagation Delay & Delay types, Transactions, Events and Transaction Scheduling, Signal Attributes.
- Design & Modeling Tools: Tutorials on available Simulators and Design Tools.
- Structural Models: Configuration Statement, Modeling Iterative/Regular Structures.
- Design Organization & Parameterization: Packages & Libraries, Design Parameterization, Design Configuration & General Purpose Test Bench.

1-9

Course Content

- Dataflow Models: Concurrent Signal Assignment, Block statements, Guards, Resolution Functions, Resolved Signals and Signal Kinds, Data Flow Moore & Mealy Models, Data & Control Path Data Flow Models.
- Behavioral Models: Process & Wait Statements, Assert Statement, General Algorithmic Model, Moore and Mealy Machine Algorithmic Models, Data & Control Path Design.
- Writing Test Benches: Types of Test Benches, Examples.
- Introduction to VHDL Synthesis: Combinational, Sequential Logic Synthesis, State Machine Synthesis. VHDL Coding Styles for Synthesis.
- CPU Design Example: Behavioral Modeling of CPU, Datapath and Control Unit Modeling, CPU-Memory Interface.

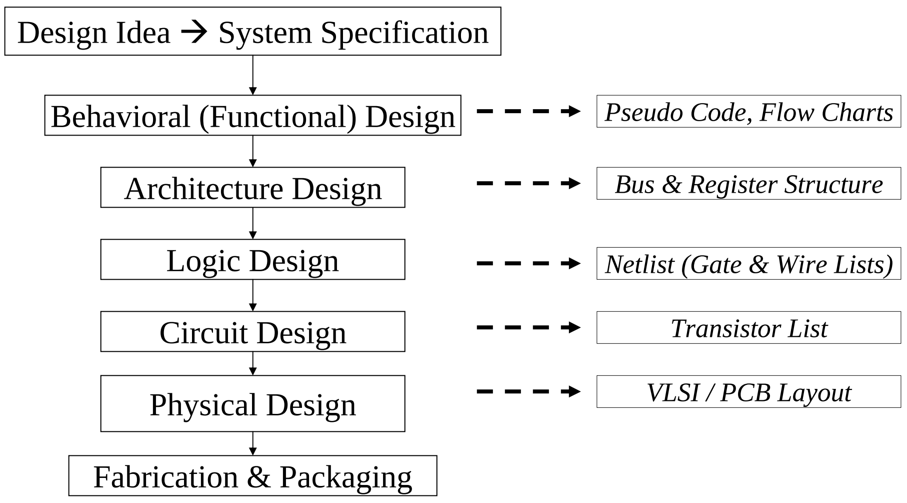
1-10

Digital System Design

- **Realization of a specification subject to the optimization of**
  - Area (Chip, PCB)
    - Lower manufacturing cost
    - Increase manufacturing yield
    - Reduce packaging cost
  - Performance
    - Propagation delay (combinational circuits)
    - Cycle time and latency (sequential circuits)
    - Throughput (pipelined circuits)
  - Power dissipation
  - Testability
    - Earlier detection of manufacturing defects lowers overall cost
  - Design time (time-to-market)
    - Cost reduction
    - Be competitive

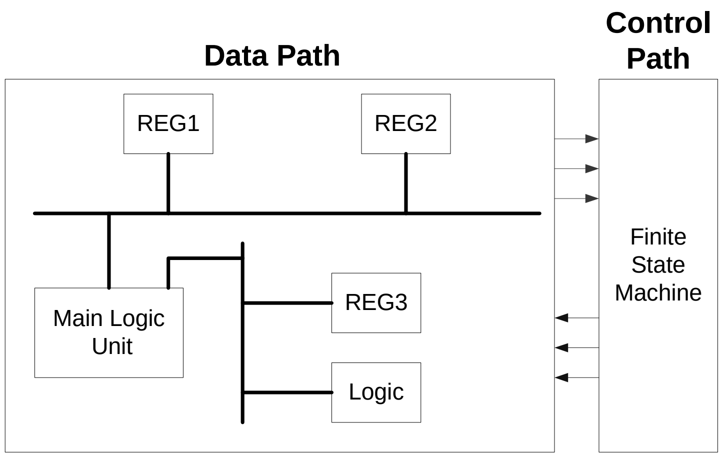
1-11

Digital System Design Cycle



1-12

Architecture Design



1-13

Architecture Design Example

- **Problem: It is required to design an 8-bit adder**
  - The two operands are stored in two 8-bit shift registers A and B
  - At the end of the addition operation, the sum must be stored in the A register.
  - The contents of the B register must not be destroyed.
  - The design must be as economical as possible in terms of hardware.

1-14

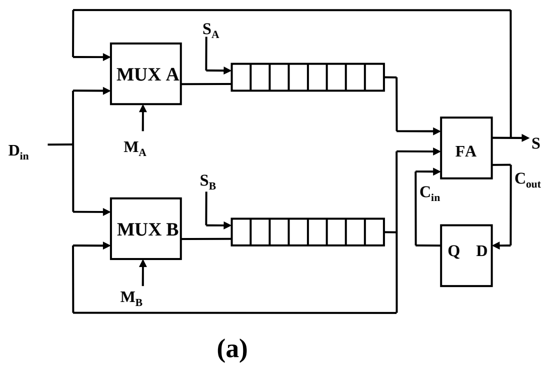
8-bit Adder Possible Solutions

- **There are numerous ways to design the above circuit, some of which are listed below.**
  - Use an 8-bit ripple-carry adder
  - Use an 8-bit carry look-ahead adder.
  - Use two 4-bit carry look-ahead adders and ripple the carry between stages.
  - Use a 1-bit adder and perform the addition serially in 8 clock cycles.

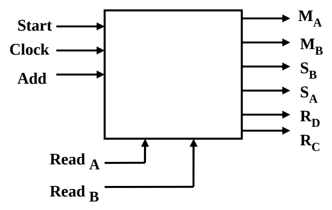
1-15

8-bit Adder Selected Design

- **Since it is specified that the hardware cost must be minimum, the serial adder design is selected.**



(a)



(b)

1-16

Data Path & Control Unit of Serial Adder

- **Data path consists of**
  - Two 8-bit shift registers
  - A full adder
  - A D-flip flop
  - Two multiplexers
  - 3-bit Counter
- **Control unit generates the following signals**
  - S_A to Shift the register A right by one bit
  - S_B to shift the register B right by one bit
  - M_A to control multiplexer A
  - M_B to control multiplexer B
  - R_D to Reset the D flip-flop
  - R_C to Reset the counter

1-17

Control Algorithm of Serial Adder

```
Forever do
  While (START = 0) skip;
     Reset the D flip-flop and the counter;
     Set $M_A$ and $M_B$ to 0;
  Repeat
     Shift registers A and B right by one
     counter = counter + 1;
  Until counter = 8;
```

1-18

Observations

- **Design involves trade-offs between**
  - Cost
  - Performance
  - Testability
  - Power dissipation
  - Fault tolerance
  - Ease of design
  - Ease of making changes to the design.
- **Serial is cheap but slow, parallel fastest in terms of performance but most costly.**
- **The different ways we can think of building an 8-bit adder constitutes what is known as *design space* (at a particular level of abstraction).**
  - Each method of implementation is called a point in the design space.

1-19

Behavioral or High-Level Synthesis

- **The automatic generation of data path and control unit is known as *high-level synthesis.***
- **Tasks involved in HLS are scheduling and allocation.**
- **Scheduling distributes the execution of operations throughout time steps.**
- **Allocation assigns hardware to operations and values.**
  - Allocation of hardware cells include functional unit allocation, register allocation and bus allocation.
  - Allocation determines the interconnections required.

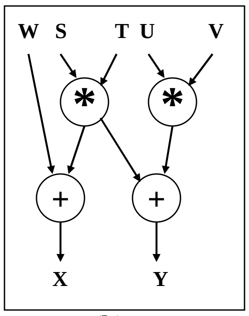
1-20

Behavioral Description and its Control Data Flow Graph (CDFG)

X = W + (S * T)
Y = (S * T) + (U * V)

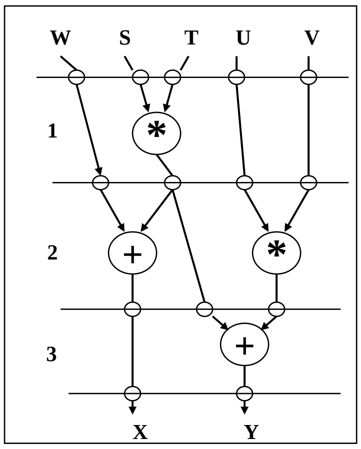
(a)

CDFG



(b)

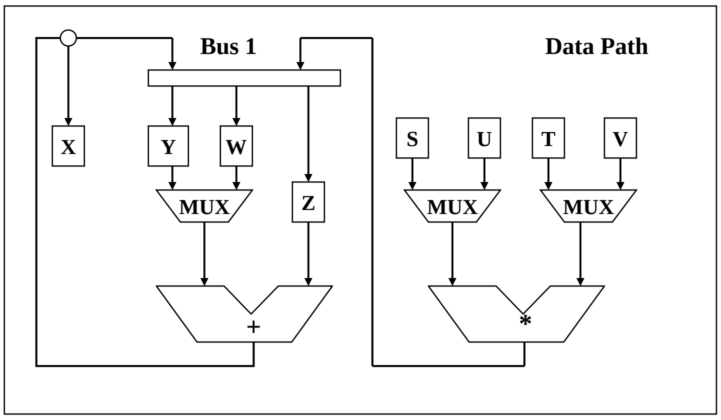
Scheduled CDFG



(c)

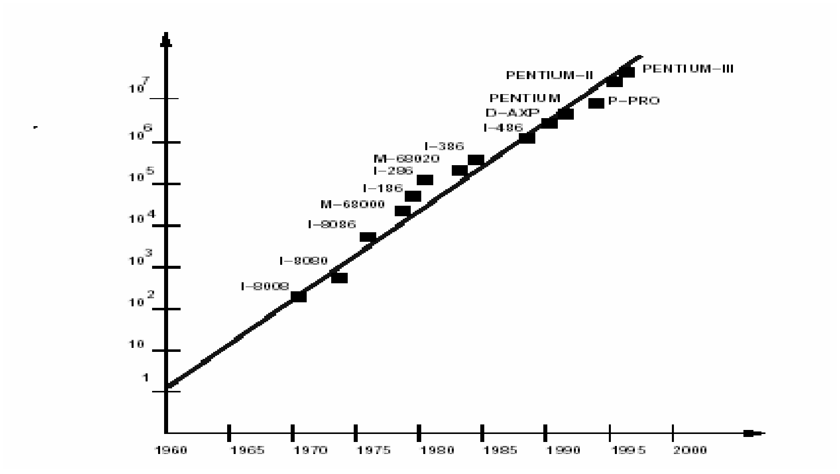
1-21

Resulting Architecture Design



1-22

Digital System Complexity



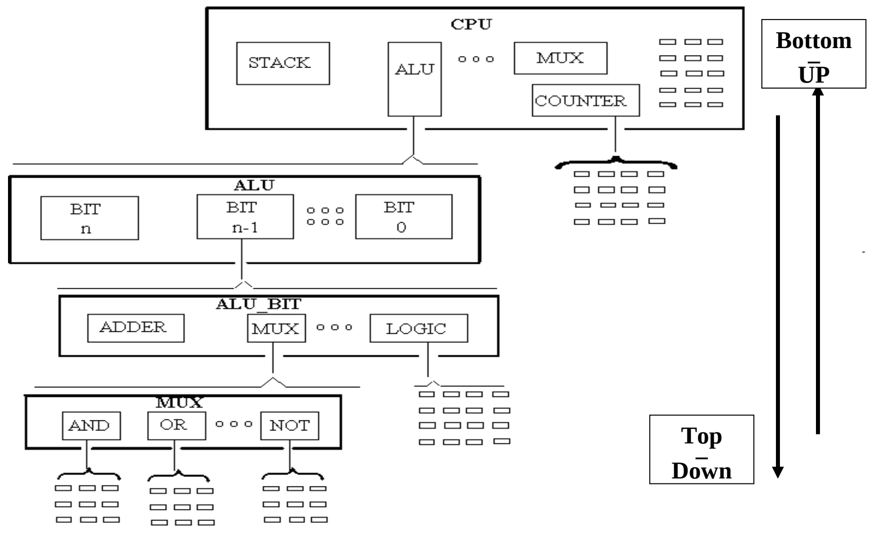
1-23

How to Deal with Design Complexity?

- **Moore's Law: Number of transistors that can be packed on a chip doubles every 18 months while the price stays the same.**
- **Hierarchy: structure of a design at different levels of description.**
- **Abstraction: hiding the lower level details.**

1-24

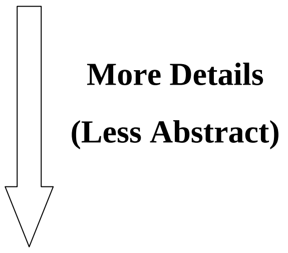
Design Hierarchy



1-25

Abstractions

- An ***Abstraction*** is a simplified model of some Entity which ***hides certain amount of the Internal details of this Entity***
- **Lower Level abstractions give more details of the modeled Entity.**
- **Several levels of abstractions (*details*) are commonly used:**
  - System Level
  - Chip Level
  - Register Level
  - Gate Level
  - Circuit (Transistor) Level
  - Layout (Geometric) Level



1-26

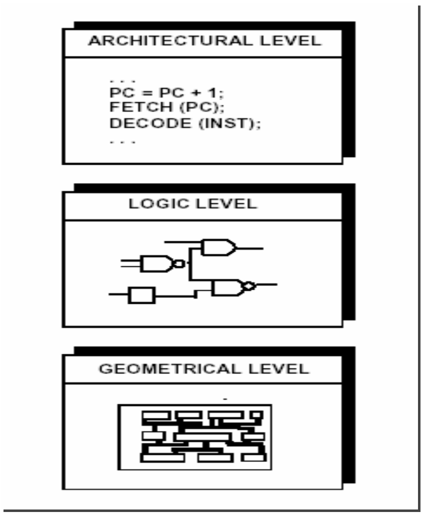
Design Domains & Levels of Abstraction

- **Designs can be expressed / viewed in one of three possible domains**
  - Behavioral Domain (***Behavioral View***)
  - Structural/Component Domain (***Structural View***)
  - Physical Domain (***Physical View***)
- **A design modeled in a given domain can be represented at several levels of abstraction (*Details*).**

1-27

Three Abstraction Levels of Circuit Representation

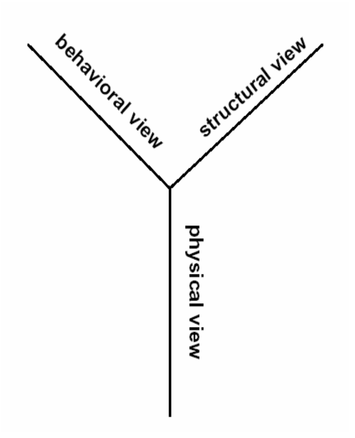
- Architectural level
  - Operations implemented by resources.
- Logic level
  - Logic functions implemented by gates.
- Geometrical level
  - Devices are geometrical objects.



1-28

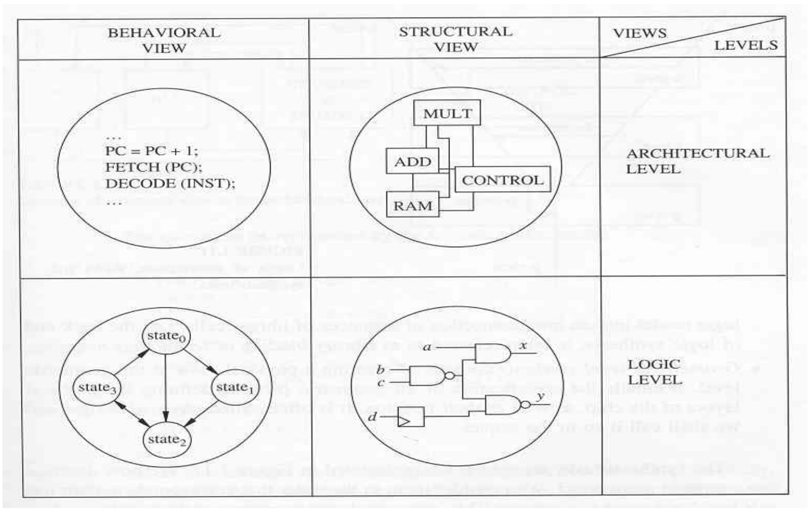
Modeling Views

- Behavioral view
  - Abstract function.
- Structural view
  - An interconnection of parts.
- Physical view
  - Physical objects with size and positions.



1-29

Levels of Abstractions & Corresponding Views



1-30

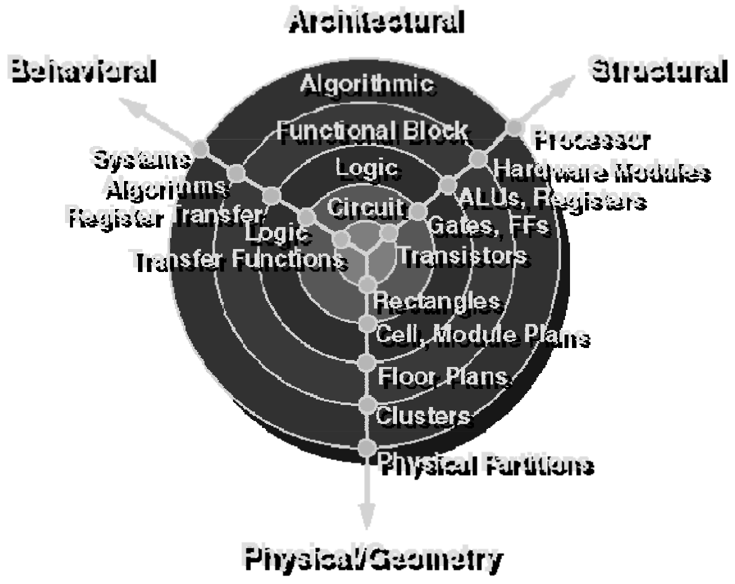
Design Domains & Levels of Abstraction

Design Domain

Abstraction Level	*Behavioral*	*Structural*	*Physical*
System	**English Specs**	**Computer, Disk Units, Radar, etc.**	**Boards, MCMs, Cabinets, Physical Partitions**
Chip	**Algorithms, Flow Charts**	**Processors, RAMs, ROMs**	**Clusters, Chips, PCBs**
Register	**Data Flow, Reg. Transfer**	**Registers, ALUs, Counters, MUX, Buses**	**Std. Cells, Floor Plans**
Gate	**Boolean Equations**	**AND, OR, XOR, FFs, etc**	**Cells, Module Plans**
Circuit (Tr)	**Diff, and element Equations**	**Transistors, R, C, etc …**	**Mask Geometry (Layout)**

1-31

Gajski and Kuhn's Y Chart



1-32

Design Methods

- **Full custom**
  - Maximal freedom
  - High performance blocks
  - Slow
- **Semi-custom**
  - Gate Arrays
    - Mask Programmable (MPGAs)
    - Field Programmable (FPGAs))
  - Standard Cells
  - Silicon Compilers & Parametrizable Modules (adder, multiplier, memories)

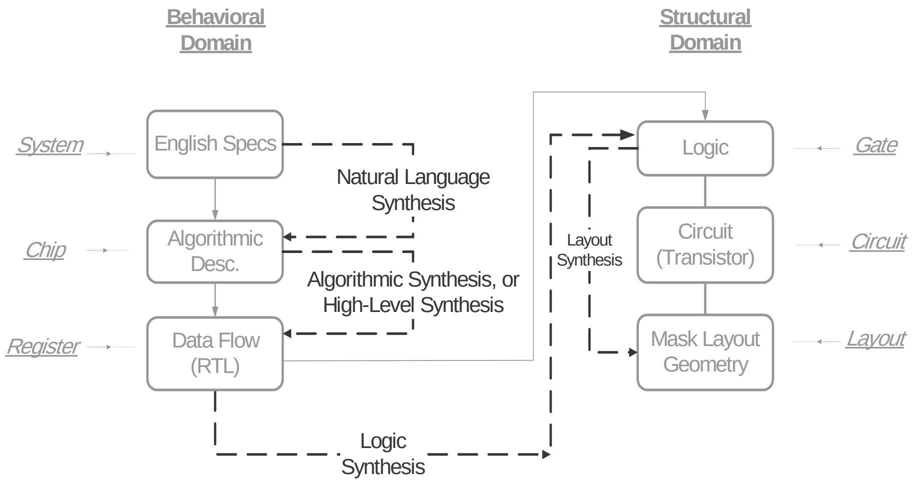
1-33

Design vs. Synthesis

- **Synthesis**
  - Process of transforming H/W from one level of abstraction to a ***lower*** one.
- **Synthesis may occur at many different levels of abstraction**
  - Behavioral or High-level synthesis
  - Logic synthesis
  - Layout synthesis
- **Design**
  - A Sequence of synthesis steps down to a level of abstraction which is manufacturable.

1-34

Synthesis Process



1-35

Circuit Synthesis

- **Architectural-level synthesis**
  - Determine the *macroscopic* structure
    - Interconnection of major building blocks.
- **Logic-level synthesis**
  - Determine the *microscopic* structure
    - Interconnection of logic gates.
- **Geometrical-level synthesis (Physical design)**
  - Placement and routing.
  - Determine positions and connections.

1-36

Design Automation & CAD Tools

- **Design Entry (Description) Tools**
  - Schematic Capture
  - Hardware Description Language (HDL)
- **Simulation (Design Verification) Tools**
  - Simulators (Logic level, Transistor Level, High Level Language "HLL")
- **Synthesis Tools**
- **Formal Verification Tools**
- **Design for Testability Tools**
- **Test Vector Generation Tools**

1-37

Hardware Description Languages

- **HDLs are used to describe the hardware for the purpose of modeling, simulation, testing, design, and documentation.**
  - Modeling: behavior, flow of data, structure
  - Simulation: verification and test
  - Design: synthesis
- **Two widely-used HDLs today**
  - **VHDL:** VHSIC (Very High Speed Integrated Circuit) Hardware Description Language (IEEE standard)
  - **Verilog** (from Cadence, now IEEE standard)

1-38

Objectives of VHDL

- **Provide a unified notation to describe Electronic Systems (digital hardware) at various levels of abstractions.**
- **Standardization of documentation**
  - To support the communication of design data
- **System design time and cost**
  - reduced ambiguity in specification of design interfaces and design functions
  - reusability of existing designs
- **Open-system CAE tools**
  - can change CAE system without losing use of existing designs
  - elimination of language translators
- **Improved integration of multi-vendor designs**
  - shared design databases become possible
  - standard cells, behavioral models

1-39

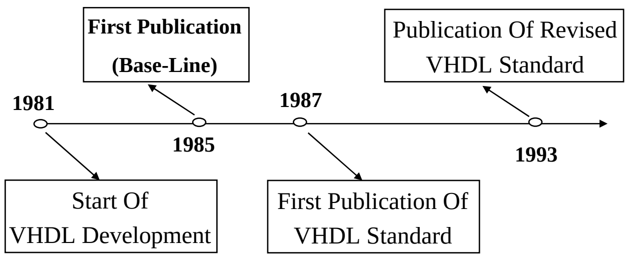
VHDL Requirements

- **Support for design hierarchy**
- **Library support**
- **Sequential statement**
- **Generic design**
- **Type declaration and usage**
- **Use of subprograms**
- **Timing control**
- **Structural specification**

1-40

History of VHDL

- **Created by DoD to document military designs for portability**
- **IEEE standard 1076 (VHDL) in 1987**
- **Revised IEEE standard 1076 (VHDL) in 1993**
- **IEEE standard 1164 (object types standard) in 1993**
- **IEEE standard 1076.3 (synthesis standard) in 1996**


First Publication (Base-Line)
Publication Of Revised VHDL Standard
1981
1987
1985
1993
Start Of VHDL Development
First Publication Of VHDL Standard

1-41

VHDL Advantages

- **Modular**
- **Hierarchical, allows design description:**
  - TOP - DOWN
  - BOTTOM - UP
- **Portable**
- **Can describe the Same design Entity using more than one view (Domain):**
  - The Behavioral View (e.g. as an algorithm, Register-Transfer (Data Flow), Input-Output Relations, etc)
  - The Structural View.
- **This allows investigation of design alternatives of the same Entity.**
- **It also allows delayed detailed Implementations.**
- **Can model systems at various levels of abstraction (System, chip RTL, Logic (Gate))**
- **VHDL can be made to simulate timing at reasonable accuracy.**

1-42

Styles in VHDL

- **Behavioral**
  - High level, algorithmic, sequential execution
  - Hard to synthesize well
  - Easy to write and understand (like high-level language code)
- **Dataflow**
  - Medium level, register-to-register transfers, concurrent execution
  - Easy to synthesize well
  - Harder to write and understand (like assembly code)
- **Structural**
  - Low level, netlist, component instantiations and wiring
  - Trivial to synthesize
  - Hardest to write and understand (very detailed and low level)

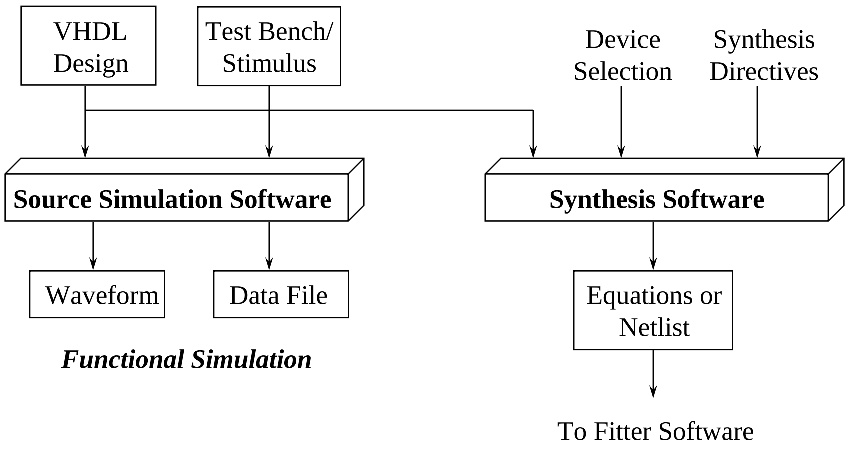
1-43

Design Flow in VHDL

- **Define the design requirements**
- **Describe the design in VHDL**
  - Top-down, hierarchical design approach
  - Code optimized for synthesis or simulation
- **Simulate the VHDL source code**
  - Early problem detection before synthesis
- **Synthesize, optimize, and fit (place and route) the design for a device**
  - Synthesize to equations and/or netlist
  - Optimize equations and logic blocks subject to constraints
  - Fit into the components blocks of a given device
- **Simulate the post-layout design model**
  - Check final functionality and worst-case timing
- **Program the device (if PLD) or send data to ASIC vendor**

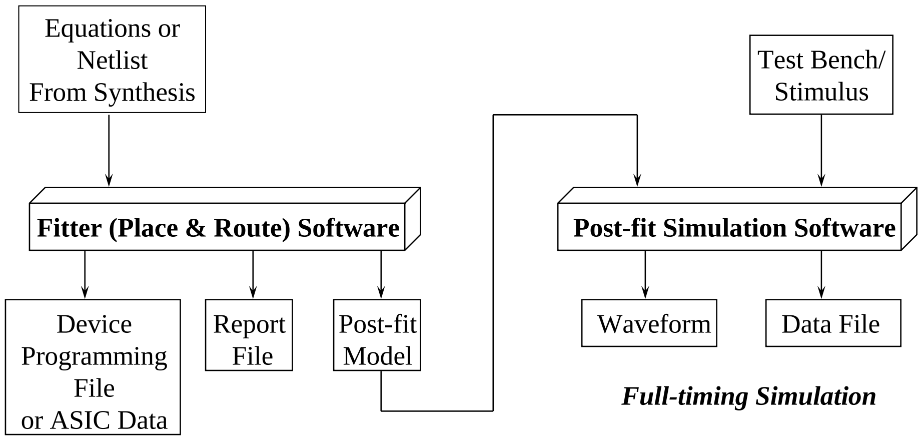
1-44

Design Tool Flow …



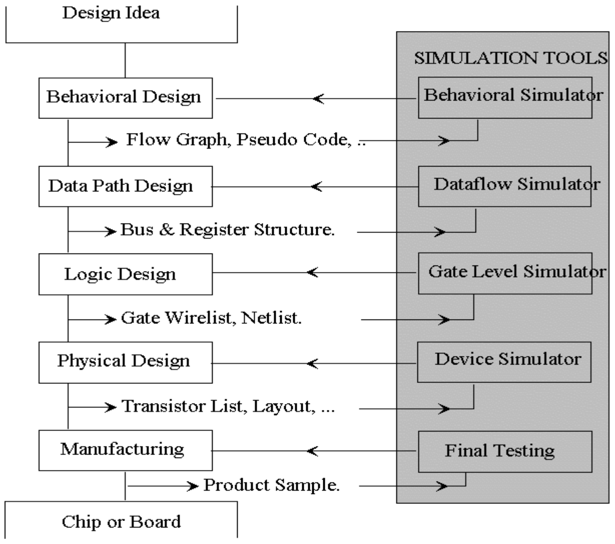
1-45

… Design Tool Flow


Equations or Netlist From Synthesis
Test Bench/ Stimulus
Fitter (Place & Route) Software
Post-fit Simulation Software
Device Programming File or ASIC Data
Report File
Post-fit Model
Waveform
Data File
Full-timing Simulation

1-46

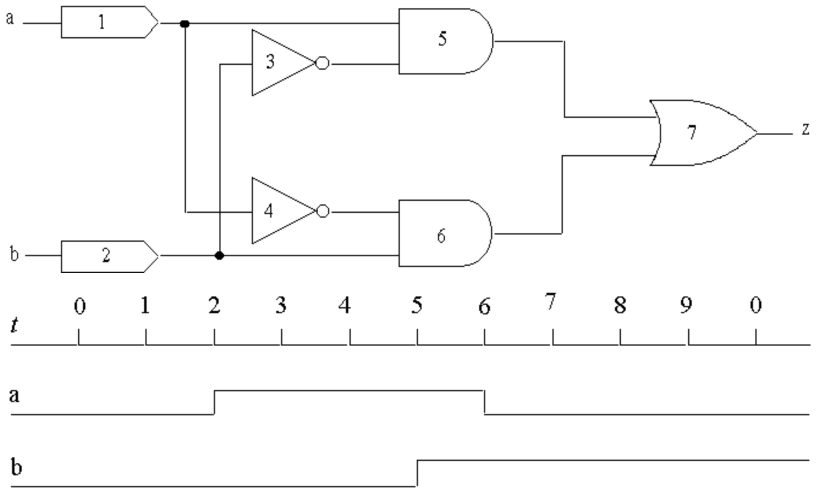
Simulation Process



1-47

Simulation Types

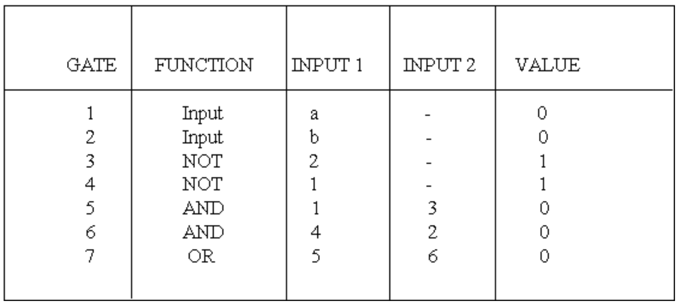
- **Oblivious and Event-driven simulation**



1-48

Oblivious Simulation

- Need a tabular netlist for oblivious simulation
- Simulate fixed time intervals
- Update table values at each interval

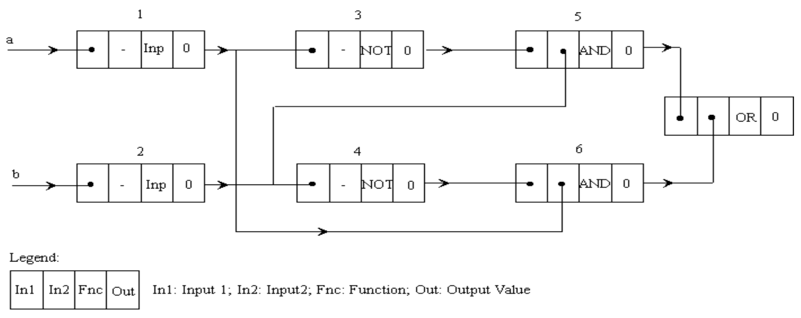


GATE	FUNCTION	INPUT 1	INPUT 2	VALUE
1	Input	a	-	0
2	Input	b	-	0
3	NOT	2	-	1
4	NOT	1	-	1
5	AND	1	3	0
6	AND	4	2	0
7	OR	5	6	0

1-49

Event-Driven Simulation

- Evaluate circuit only when events occur
- Offers a faster simulation for digital systems
- VHDL is an event driven simulation



1-50