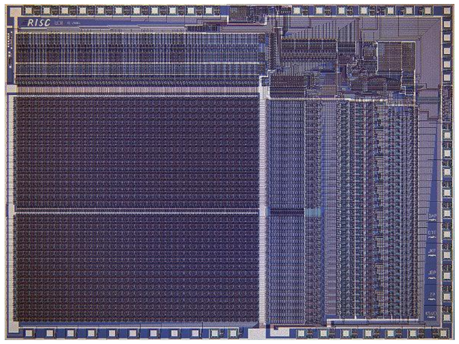
EE 260: Introduction to Digital Design Overview

Yao Zheng
Department of Electrical Engineering
University of Hawai‘i at Mānoa

1

Class Objectives

- Fundamentals of logic design
- Understand the functionalities of TTL devices
- Basic usage of Hardware Design Language
- Design small-to-moderate digital systems



Basic sequential processor
~50,000 Transistors

18

Photo of Berkeley RISC I, © University of California (Berkeley)

Class Logistic

- Lectures (50 minutes)
- Recommended Readings
- Problem Sets
- Labs
- Midterm (2)
- Final Exam (or Project)

Class Logistic

- **Lecturer:**
  - Yao Zheng (yao.zheng@hawaii.edu)
- **TA:**
  - Joshua Chen (jschen2@hawaii.edu)
- **Grader:**
  - Nicholas Yama (nyama8@hawaii.edu)

Class Logistic

- **Lecture**:
  Mondays, Wednesdays, and Fridays 9:30am-10:20am, Marine Science Building 100.
- **Laboratory**:
  Tuesdays 9:00am-11:45am, Holmes Hall 451.
- **Office hours**:
  Mondays and Wednesdays 14:00pm-16:00pm or email for appointment, Holmes Hall 437.

Class Logistic

- Course website:
  - http://www2.hawaii.edu/~yaozheng/course/ee260_spring_2018/
- Piazza:
  - https://piazza.com/hawaii/spring2018/ee260
- Github Classroom
  - https://ee260-2018-spring-signup.herokuapp.com

Course Logistic

- **Textbook**:
  Required: Digital Design Principles and Practice with Verilog (5th Edition):
- Interactive Digital Access Program (through Laulima)
- Bookstore

- Optional: FPGA Prototyping By Verilog Examples: Xilinx Spartan-3 Version
- Amazon

Course Logistic

- **Grading**:

Homework	25%
Discussion	10%
Midterms (2)	20%
Final Exam	20%
Lab Work	25%

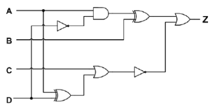
Course Logistic

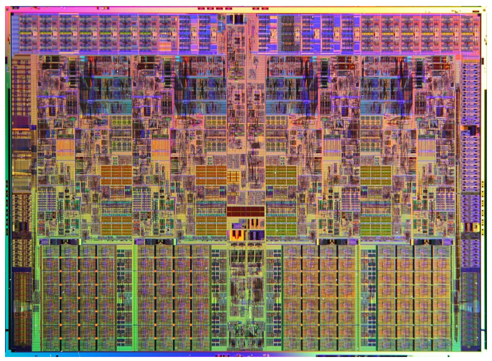
- **Basic Topics**:
  - Boolean algebra
  - Combinational logic
  - Sequential building blocks
  - Finite state machine
- **Advanced Topics:**
  - Arithmetic structures
  - Pipeline
  - Cache and Memory
- **Hardware Description Language**
  - Verilog

Know What you are Getting into

The state-of-the-art

The materials in this class





~700,000,000 Transistors

What is Computer Design?

Application

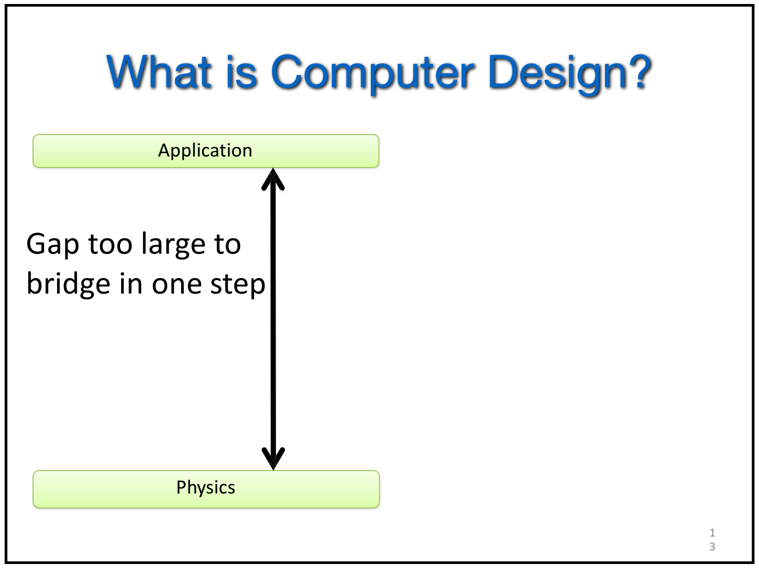
1
1

What is Computer Design?

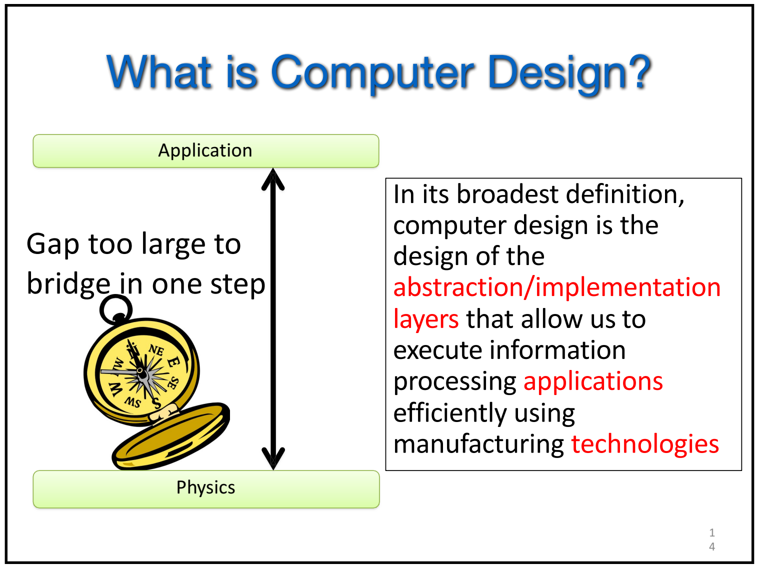
Application

Physics

1
2

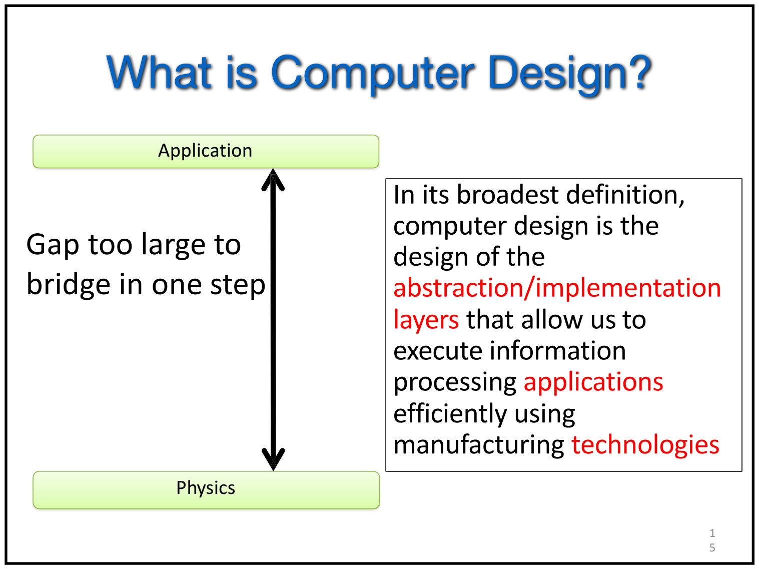


What is Computer Design?



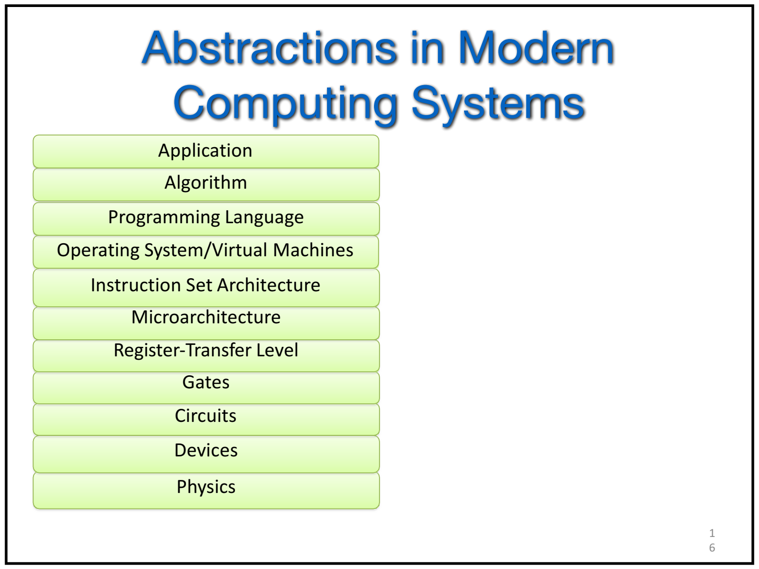
What is Computer Design?

In its broadest definition, computer design is the design of the abstraction/implementation layers that allow us to execute information processing applications efficiently using manufacturing technologies



What is Computer Design?

In its broadest definition, computer design is the design of the abstraction/implementation layers that allow us to execute information processing applications efficiently using manufacturing technologies



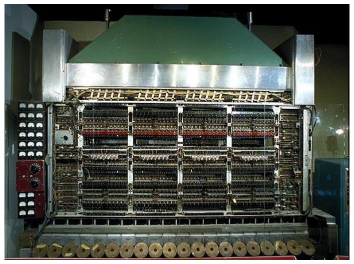
Abstractions in Modern Computing Systems

Abstractions in Modern Computing Systems

- Application
- Algorithm
- Programming Language
- Operating System/Virtual Machines
- Instruction Set Architecture
- Microarchitecture
- Register-Transfer Level
- Gates
- Circuits
- Devices
- Physics

17

What is an Application

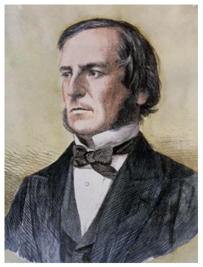


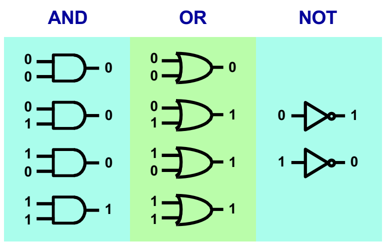
The Babbage Difference Engine (1834)
25,000 parts
cost: £17,470

- **The first digital systems were mechanical and used base-10 representation.**
- **Most popular applications: arithmetic and scientific computation**

18

Why Digital

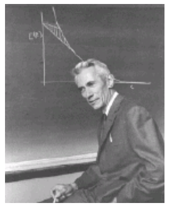


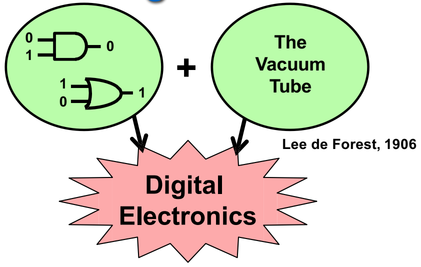


- **1854: George Boole shows that logic is math, not just philosophy!**
- **Boolean algebra: the mathematics of binary values**

19

Key Link Between Logic and Circuits


Claude Shannon
Courtesy of Jonah Sacks. Used with permission.

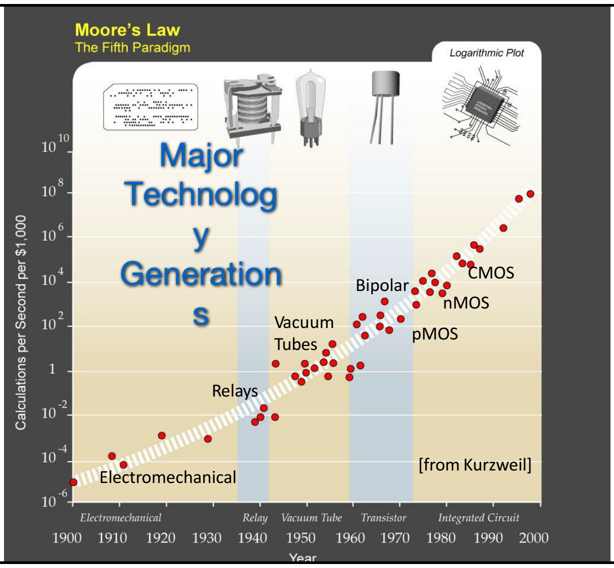


- **Despite existence of relays and introduction of vacuum tube in 1906 *digital* electronics did not emerge for thirty years!**
- **Claude Shannon notices similarities between Boolean and electronic telephone switches**
- **Shannon's 1937 MIT Master's Thesis introduces the world to binary digital electronics**

Evolution of Digital Electronics

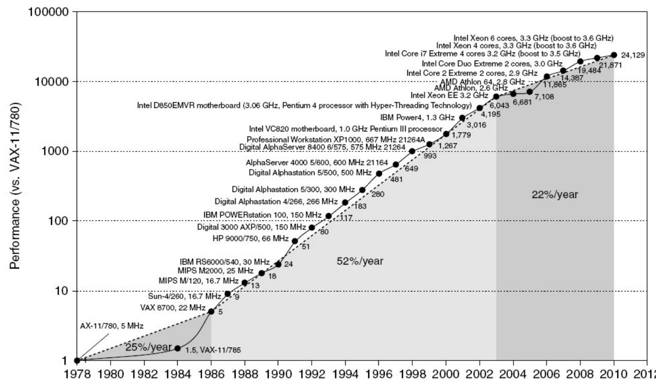
Vacuum Tubes	Transistors	VLSI Circuits
ENIAC, 1946	First Transistor Bell Labs, 1948	4004, 1971
UNIVAC, 1951	**IBM System/360, 1964**	**Intel Itanium, 2003**
1900 adds/sec	500,000 adds/sec	2,000,000,000 adds/sec

Major Technology Generations



13

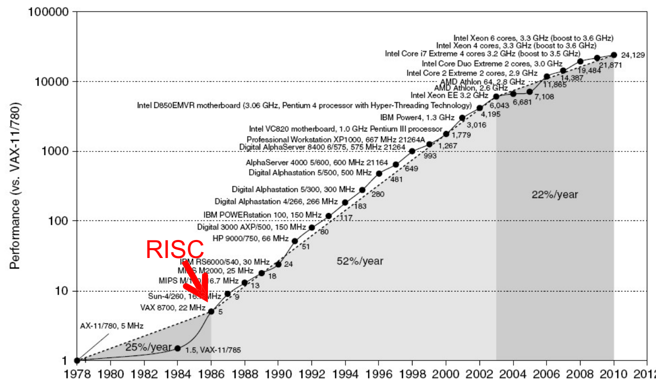
Sequential Processor Performance



23

From Hennessy and Patterson Ed. 5 Image Copyright © 2011, Elsevier Inc. All rights Reserved.

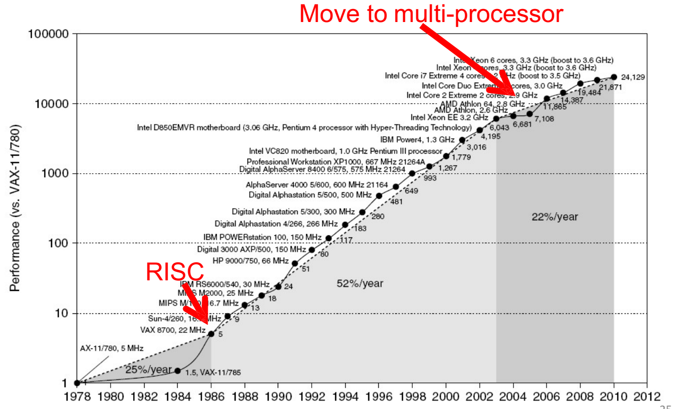
Sequential Processor Performance



24

From Hennessy and Patterson Ed. 5 Image Copyright © 2011, Elsevier Inc. All rights Reserved.

Sequential Processor Performance

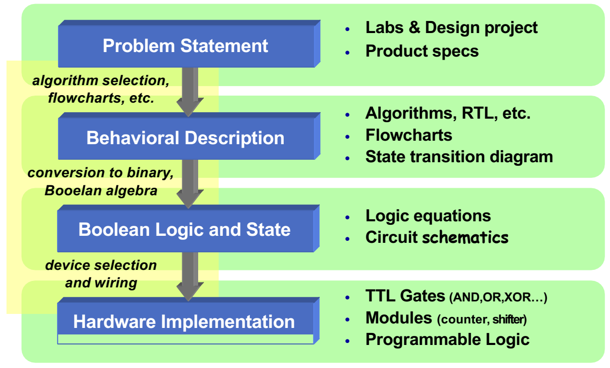


25

From Hennessy and Patterson Ed. 5 Image Copyright © 2011, Elsevier Inc. All rights Reserved.

Building Digital Systems

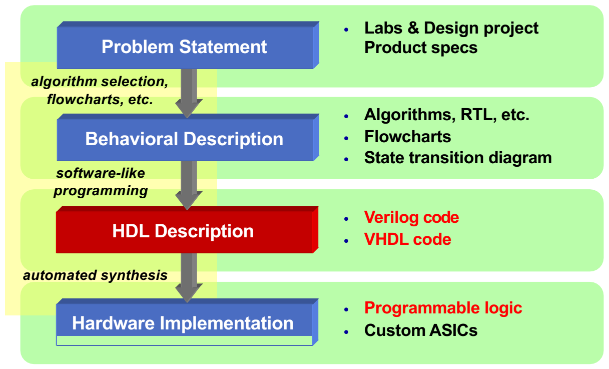
- Goal: Building binary digital solutions to computational problems


Problem Statement
algorithm selection, flowcharts, etc.
Behavioral Description
conversion to binary, Booelan algebra
Boolean Logic and State
device selection and wiring
Hardware Implementation

- Labs & Design project
- Product specs
- Algorithms, RTL, etc.
- Flowcharts
- State transition diagram
- Logic equations
- Circuit schematics
- TTL Gates (AND,OR,XOR…)
- Modules (counter, shifter)
- Programmable Logic

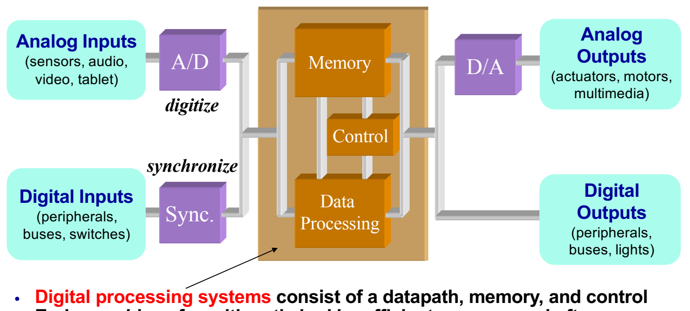
Building Digital Systems

- Logic synthesis using a Hardware Description Language (HDL) automates the most tedious and error-prone aspects of design



- Labs & Design project
- Product specs
- Algorithms, RTL, etc.
- Flowcharts
- State transition diagram
- Verilog code
- VHDL code
- Programmable logic
- Custom ASICs

Digital System Model


Analog Inputs
(sensors, audio, video, tablet)
A/D
digitize
Digital Inputs
(peripherals, buses, switches)
synchronize
Sync.
Memory
Control
Data Processing
D/A
Analog Outputs
(actuators, motors, multimedia)
Digital Outputs
(peripherals, buses, lights)

- Digital processing systems consist of a datapath, memory, and control Early machines for arithmetic had insufficient memory, and often depended on users for control
- Today's digital systems are increasingly embedded into everyday places and things
- Richer interaction with the user and environment

Digital System Model

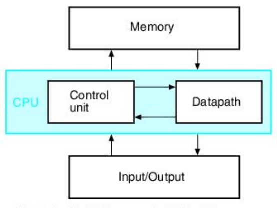


Fig. 1-2 Block Diagram of a Digital Computer

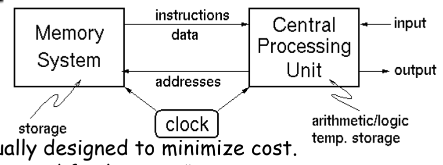
Memory: stores programs, i/o data
Datapath: perform arithmetic and other data-processing operations
Control Unit: supervises the flow of information
I/O: places program into memory
CPU: executes program from memory instruction by instruction. Control unit manipulates datapath to execute instructions.

Processor: contains CPU, FPU and MMU
FPU (Floating Point Unit) : performs floating-point operations
MMU (Memory Management Unit): helps instruction fetching and data I/O. Internal/external cache (very fast), RAM (between disk and cache)

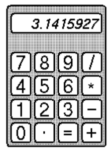
29

Example Digital Systems

- Digital Computer
  - Usually design to maximize performance. "Optimized for speed"



- Handheld Calculator

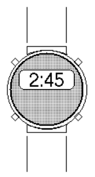


- Usually designed to minimize cost.
"Optimized for low cost"

- Of course, low cost comes at the expense of speed.

Example Digital Systems

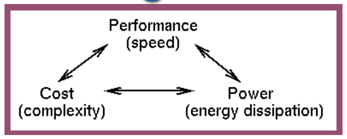
- Digital Watch



- Low p
the expense of:

Designed to minimize power.
Single battery must last for years.

Basic Design Tradeoffs



- You can improve on one at the expense of worsening one or both of the others.
- These tradeoffs exist at every level in the system design - every sub-piece and component.
- Design Specification -
  - Functional Description.
  - Performance, cost, power constraints.
- As a designer you must make the tradeoffs necessary to achieve the function within the constraints.

Cost vs Speed vs Energy

AMD Phenom X4
- X86 Instruction Set
- Quad Core
- 125W
- Decode 3 Instructions/Cycle/Core
- 64KB L1 I Cache, 64KB L1 D Cache
- 512KB L2 Cache
- Out-of-order
- 2.6GHz

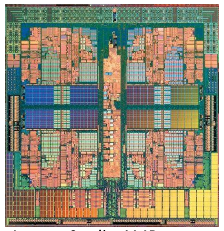


Image Credit: AMD

Intel Atom
- X86 Instruction Set
- Single Core
- 2W
- Decode 2 Instructions/Cycle/Core
- 32KB L1 I Cache, 24KB L1 D Cache
- 512KB L2 Cache
- In-order
- 1.6GHz

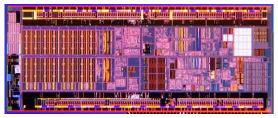


Image Credit: Intel

35

Cost vs Speed vs Energy

AMD Phenom X4
- X86 Instruction Set
- Quad Core
- 125W
- Decode 3 Instructions/Cycle/Core
- 64KB L1 I Cache, 64KB L1 D Cache
- 512KB L2 Cache
- Out-of-order
- 2.6GHz

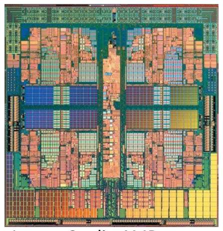


Image Credit: AMD

IBM POWER7
- Power Instruction Set
- Eight Core
- 200W
- Decode 6 Instructions/Cycle/Core
- 32KB L1 I Cache, 32KB L1 D Cache
- 256KB L2 Cache
- Out-of-order
- 4.25GHz

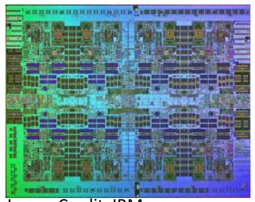
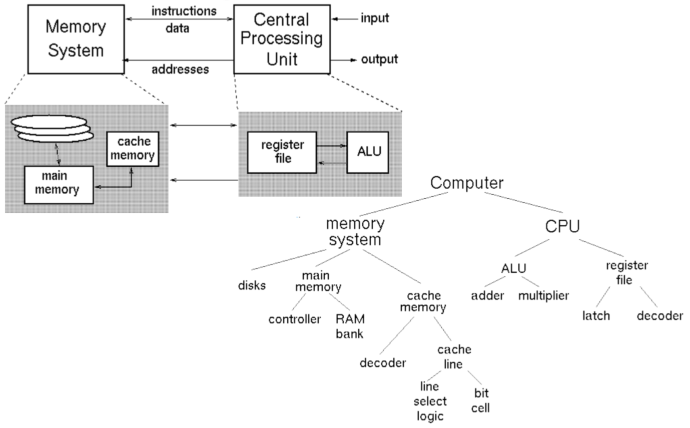


Image Credit: IBM

Courtesy of International Business Machines Corporation, © International Business Machines Corporation.

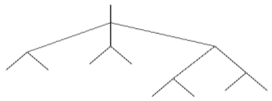
36

Design Representation



Hierarchy in Designs

- Helps control complexity -
  - by hiding details and reducing the total number of things to handle at any time.
- Modulalizes the design -
  - divide and conquer
  - simplifies implementation and debugging
- Top-Down Design
  - Starts at the top (root) and works down by successive refinement.
- Bottom-up Design
  - Starts at the leaves & puts pieces together to build up the design.
- Which is better?
  - In practice both are needed & used.
    - Need top-down divide and conquer to handle the complexity.
    - Need bottom-up because in a well designed system, the structure is influence by what primitves are available.



Software Aspects of Design

- In computer-aided design (CAD) various tools are available today to improve productivity, correctness and the quality of designs.
- Ways of digital design in CAD:
  - Schematic entry: schematic diagrams to be drawn on screen.
  - HDLs : Hardware description languages
  - Simulators: debug and check

37

Verilog and VHDL

VHDL	Verilog
Commissioned in 1981 by Department of Defense; now an IEEE standard	Created by GatewayDesign Automation in 1985; now an IEEE standard
Initially created for ASIC synthesis	Initially an interpreted language for gate-level simulation
Strongly typed;potentially verbose code	Less explicit typing (e.g., compiler will pad arguments of different widths)
Strong support for package management and large designs	No special extensions for large designs

Hardware structures can be modeled effectively in either VHDL and Verilog. Verilog is similar to c and a bit easier to learn.

3

Verilog

- **Behavioral or Algorithmic Level**
  - Highest level in the Verilog HDL
  - Design specified in terms of algorithm (functionality) without hardware details. Similar to "c" type specification
  - Most common level of description
- **Dataflow Level**
  - The flow of data through components is specified based on the idea of how data is processed
- **Gate Level**
  - Specified as wiring between logic gates
  - Not practical for large examples
- **Switch Level**
  - Description in terms of switching (modeling a transistor)
  - No useful in general logic design – we won't use it

A design mix and match all levels in one design is possible.
In general Register Transfer Level (RTL) is used for a combination of Behavioral and Dataflow descriptions

Verilog

- **Misconceptions**
  - The coding style or clarity does not matter as long as it works
  - Two different Verilog encodings that simulate the same way will synthesize to the same set of gates
  - Synthesis just can't be as good as a design done by humans
    - Shades of assembly language versus a higher level language
- **What can be Synthesized**
  - Combinational Functions
    - Multiplexors, Encoders, Decoders, Comparators, Parity Generator Adders, Subtractors, ALUs, Multipliers
    - Random logic
  - Control Logic
    - FSMs
- **What can't be Synthesized**
  - Precise timing blocks (e.g., delay a signal by 2ns)
  - Large memory blocks (can be done, but very inefficient)

Understand what constructs are used in simulation vs. hardware mapping

Electronic Aspects

- Digital circuits deal with voltages and currents and are built with analog components. We perform ***digital***

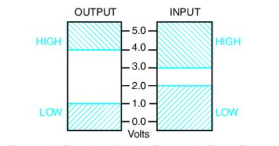
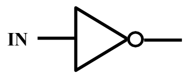


Fig. 1-1 An Example of Voltage Ranges for Binary Signals

- Electronic circuit design ensures that logic gates produce and recognize logic signals that are within the appropriate range.

41

Electronic Devices are not Ideal

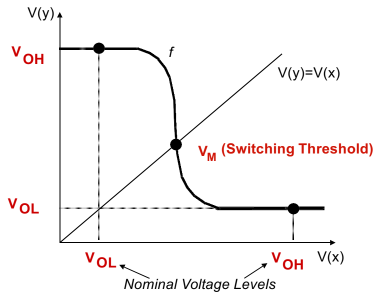


Truth Table

IN	OUT
0	1
1	0

Digital circuits perform operations on logical (or Boolean) variables

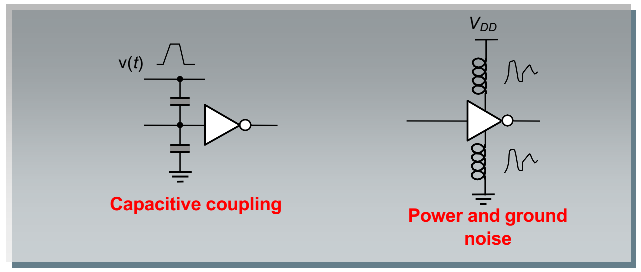
A logical variable is a mathematical abstraction. In a physical implementation, such a variable is represented by an electrical quantity



$V_{OH} = f(V_{OL})$
$V_{OL} = f(V_{OH})$
$V_M = f(V_M)$

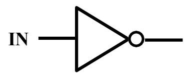
42

Example Noise Sources in Digital Circuits



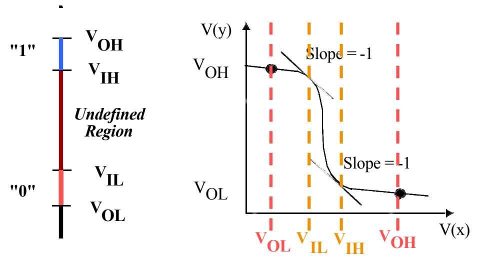
- **Noise sources: coupling, cross talk, supply noise, etc**
- **Digital circuits must be robust against such noise sources**

The Inverter: Noise Margin



Truth Table

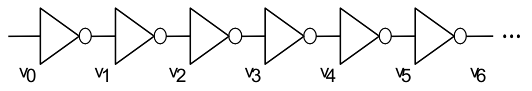
IN	OUT
0	1
1	0

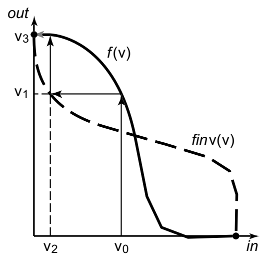


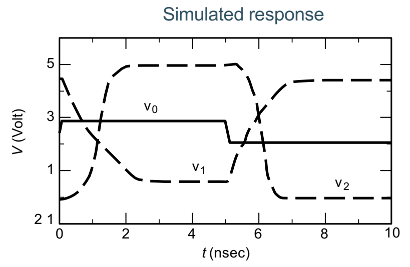
$NM_L = V_{IL} - V_{OL}$
$NM_H = V_{OH} - V_{IH}$

- Large noise margins **protect against various noise sources**

Regenerative Property


A chain of inverters




Simulated response

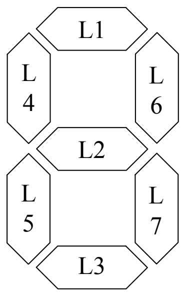
| Voltage gain | should be > 1 between logic states

Verification and Testing

- Design can be fun. Verification/testing is hard work.
- Verification by simulation (and formally through test benches) is a critical part of the design process.
- The physical hardware must be tested to debug tested mapping process and manufacturing defects.
- Physical realizations often do not allow access to internal signals. Need special tool to observe and control internal state.

Verification and Design for Test (DFT) are important components of digital design

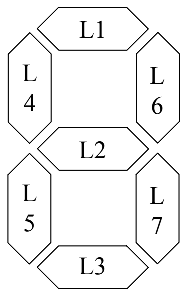
Case Study of a Simple Logic Design: Seven Segment Display



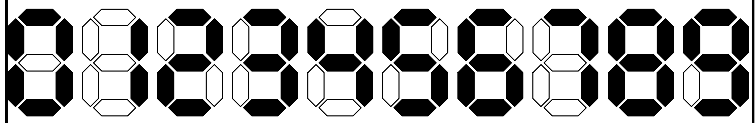
B3	B2	B1	B0	Val
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

- Chip to drive digital display

Case Study (cont.)



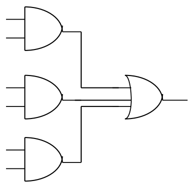
B3	B2	B1	B0	Val	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	1	1	0	1	1



Case Study (cont.)

- Implement L4:

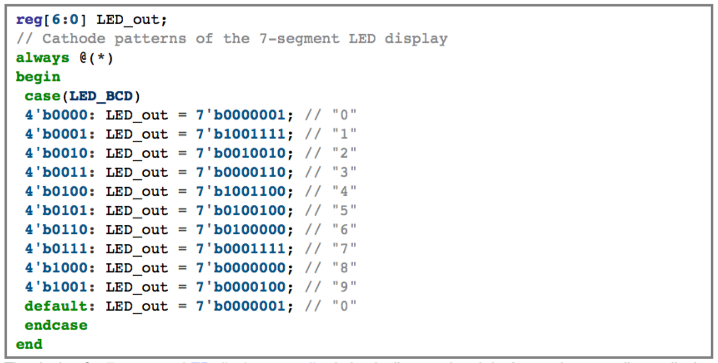
B3	B2	B1	B0	L4
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1



Some gate level implementation of the Boolean function for L4

Case Study (cont.)

- Verilog Code:



```
reg[6:0] LED_out;
// Cathode patterns of the 7-segment LED display
always @(*)
begin
 case(LED_BCD)
 4'b0000: LED_out = 7'b0000001; // "0"
 4'b0001: LED_out = 7'b1001111; // "1"
 4'b0010: LED_out = 7'b0010010; // "2"
 4'b0011: LED_out = 7'b0000110; // "3"
 4'b0100: LED_out = 7'b1001100; // "4"
 4'b0101: LED_out = 7'b0100100; // "5"
 4'b0110: LED_out = 7'b0100000; // "6"
 4'b0111: LED_out = 7'b0001111; // "7"
 4'b1000: LED_out = 7'b0000000; // "8"
 4'b1001: LED_out = 7'b0000100; // "9"
 default: LED_out = 7'b0000001; // "0"
 endcase
end
```