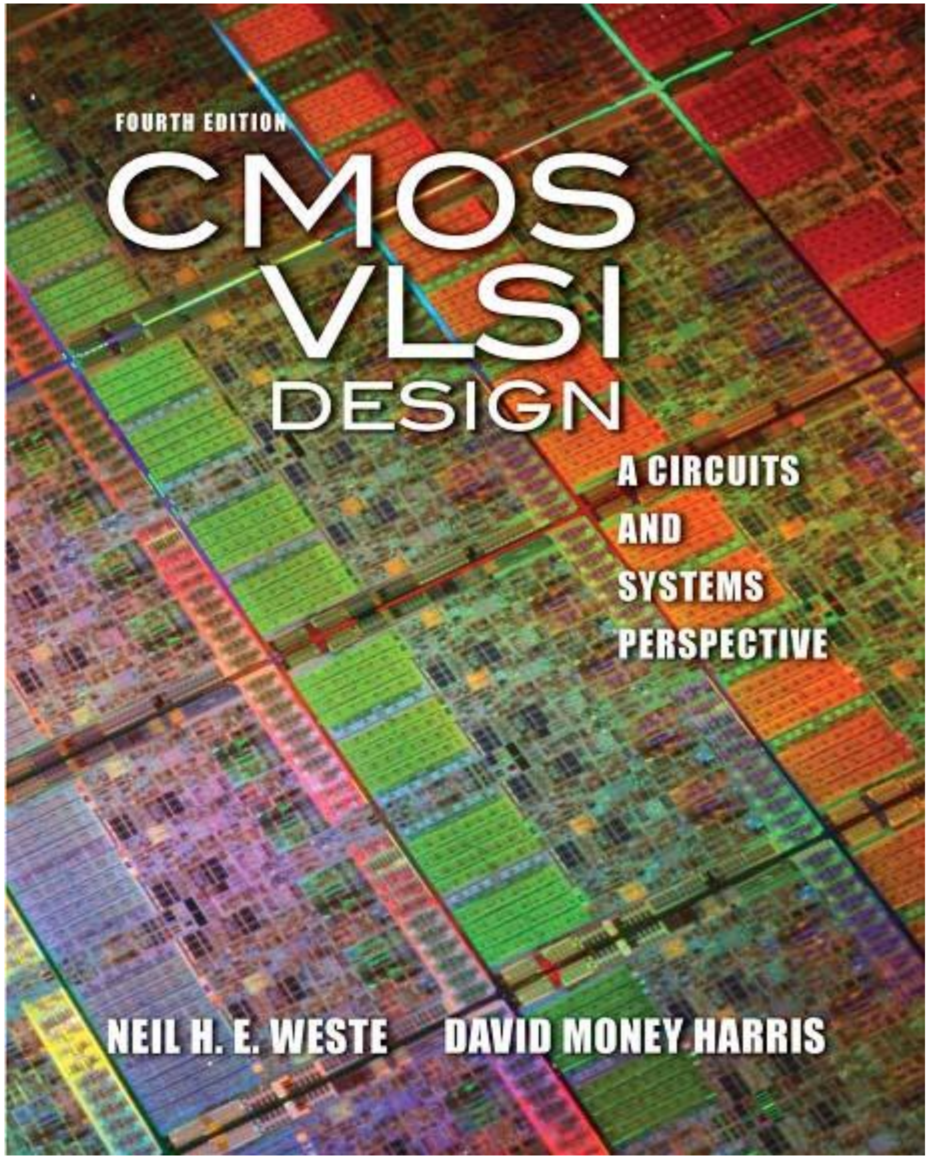
CPE 110408423 VLSI Design Chapter 14: Design Methodology Tools

Bassam Jamil

[Computer Engineering Department,

Hashemite University]



Lecture 8: Design Methodology Tools

Outline

- Introduction
- Structure Design Strategies
- Design Methods
- Design Flows
- Design Economics

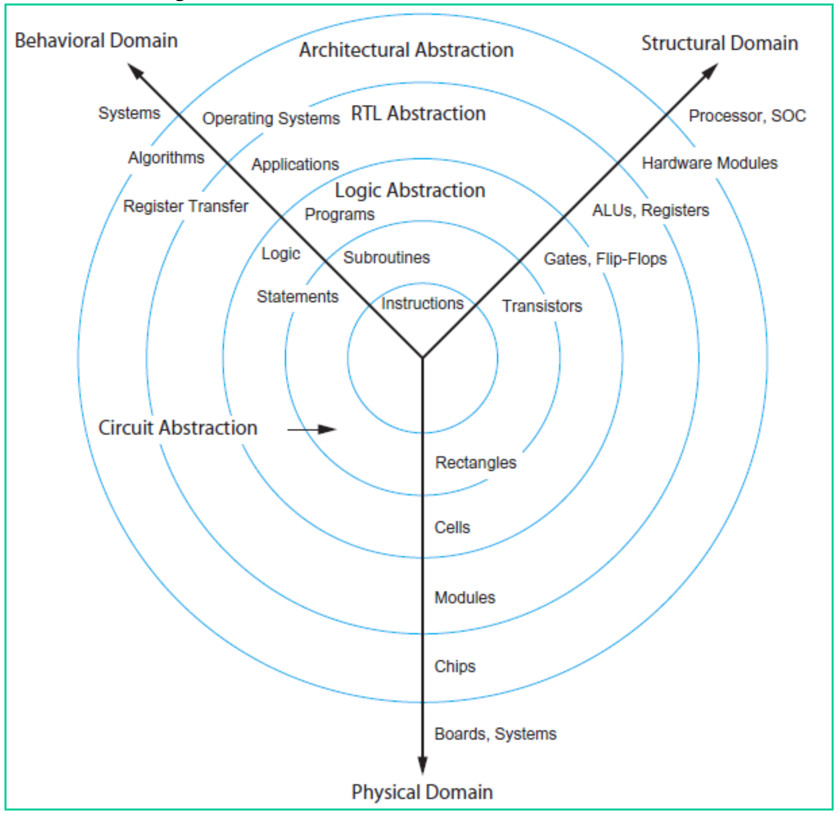
14.1 Introduction

- **Design Methodologies** are: developed and adapted strategies to form a cohesive set of principles to increase the likelihood of timely, successful designs.
- Integrated circuit can be described in terms of three domains:
  - **The behavioural domain** specifies what we wish to accomplish with a system.
  - **The structural domain** specifies the interconnection of components required to achieve the behavior we desire.
  - **The physical domain** specifies how to arrange the components in order to connect them, which in turn allows the required behavior.
- Design flows from behaviour to structure and to a physical implementation via transformations where the correctness is tested by comparing the pre- and post-transformation design.
- These domains can further be hierarchically divided into different levels of *design abstraction:*
  - Architectural or functional level
  - Logic or Register Transfer Level (RTL)
  - Circuit level

14.1 Introduction: Y-chart

- The relationship between description domains and levels of abstraction is elegantly shown by the *Gajski-Kuhn Y chart.*

- The three radial lines represent the behavioral, structural, and physical domains.
- As we move out along any of the radial axes, the increasing level of design abstraction is able to represent greater complexity.
- Circles represent levels of similar design abstraction: the architectural, RTL, logic, and circuit levels.



14.2 Structured Design Strategies

- ❑ A good VLSI design system should provide for consistent descriptions in all three description domains (behavioural, structural, and physical) and at all relevant levels of abstraction (e.g., architecture, RTL/block, logic, and circuit).
- ❑ The performance can be measured by the following design parameters:
  - – Performance—speed, power, function, flexibility
  - – Size of die (hence, cost of die)
  - – Time to design (hence, cost of engineering and schedule)
  - – Ease of verification, test generation, and testability (hence, cost of engineering and schedule).
- ❑ Design is a continuous trade-off to achieve adequate results for all of the above parameters.
- ❑ A good VLSI-design aids is to reduce complexity, increase productivity, and assure the designer of a working product.

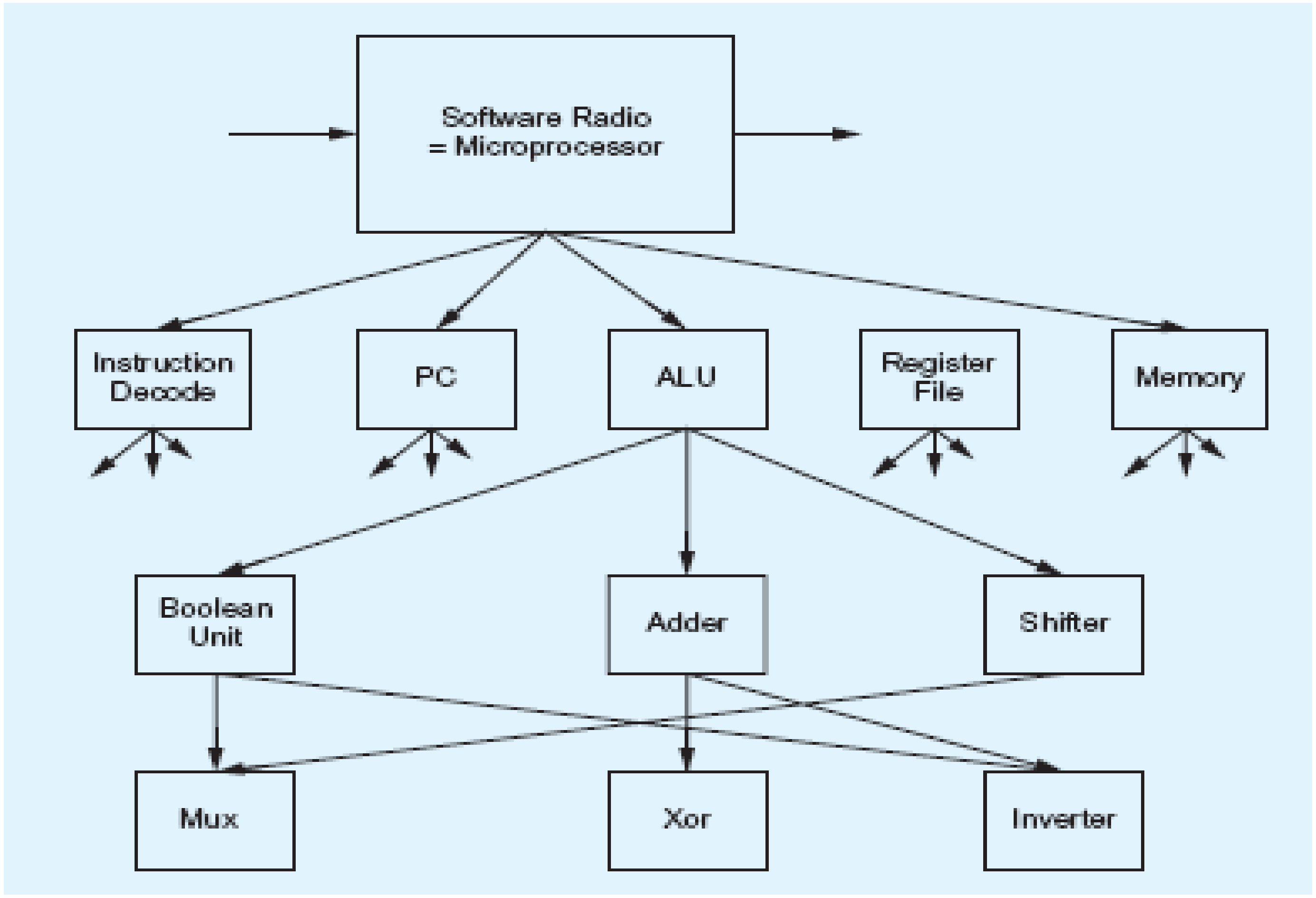
Design Principles:

- hierarchy,
- regularity,
- modularity,
- and locality

14.2.2 Hierarchy : “divide and conquer”

- ❑ Dividing a system into modules, then repeating this process until complexity of the sub modules is at an appropriately comprehensible level of detail.
- ❑ One can progress in verification from the bottom to the top of a design, checking each level of hierarchy where domains are intended to correspond.
- ❑ Hierarchy allows the use of *virtual components, soft versions of the more conventional* packaged IC. They are placed into a chip design as pieces of code. They can be supplied by an independent *intellectual property (IP) provider or can be reused from a previous product developed* in your organization.

hierarchy of software radio using a single microprocessor



14.2.3 Regularity

- The designer attempts to divide the hierarchy into a set of similar building blocks. Regularity can exist at all levels of the design hierarchy.
- Regularity aids in verification efforts by reducing the number of subcomponents to validate and by allowing formal verification programs to operate more efficiently.
- Design reuse depends on the principle of regularity to use the same virtual component in multiple places or products.

14.2.4 Modularity

- Modules have well-defined functions and interfaces.
- In the IC case, this corresponds to a clearly defined behavioural, structural, and physical **interface** that indicates the function, the name, signal type, electrical and timing constraints of the ports on the design.
- The ability to divide the task into a set of well-defined modules also aids in System-On-Chip (SOC) designs where a number of IP sources have to be interfaced to complete a design.

14.2.5 Locality

- The internals of the module are unimportant to other modules.
  - In this way we are performing a form of “information hiding” that reduces the apparent complexity of the module.
- Increasingly, locality often means temporal locality or adherence to a clock or timing protocol.
  - Reference all signals to a clock
  - Input signals are specified with required setup and hold times relative to the clock, and outputs have delays related to the edges of the clock.

14.2.6 Design Principles Summary

- There are strong parallels between the methods of design for software and hardware systems. The table summarizes some of these parallels for the principles outlined above.

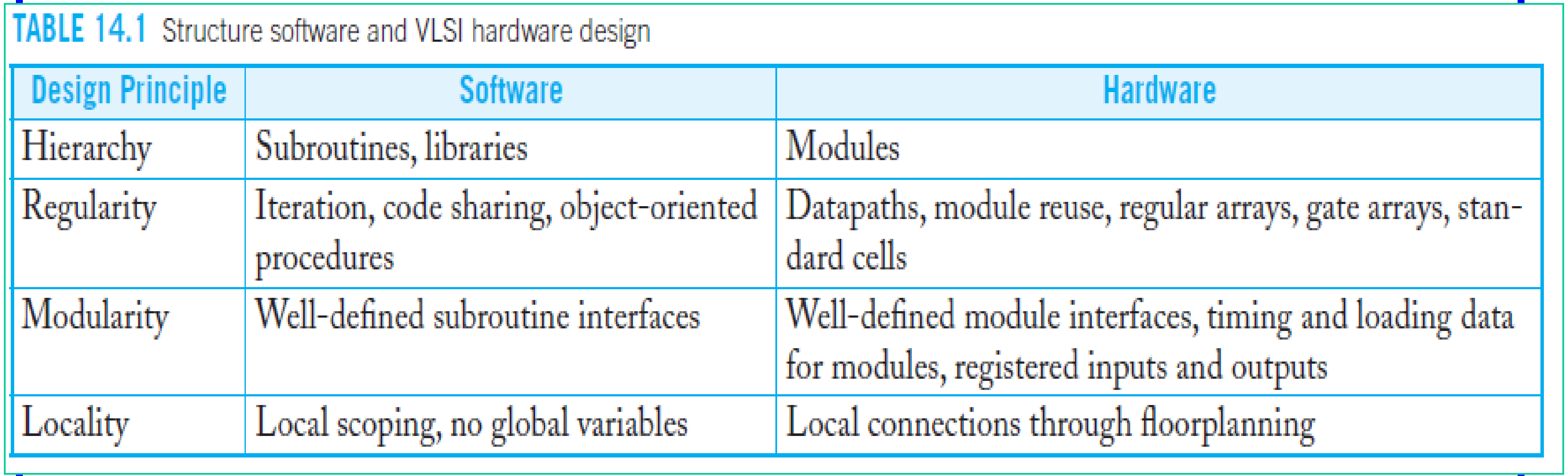
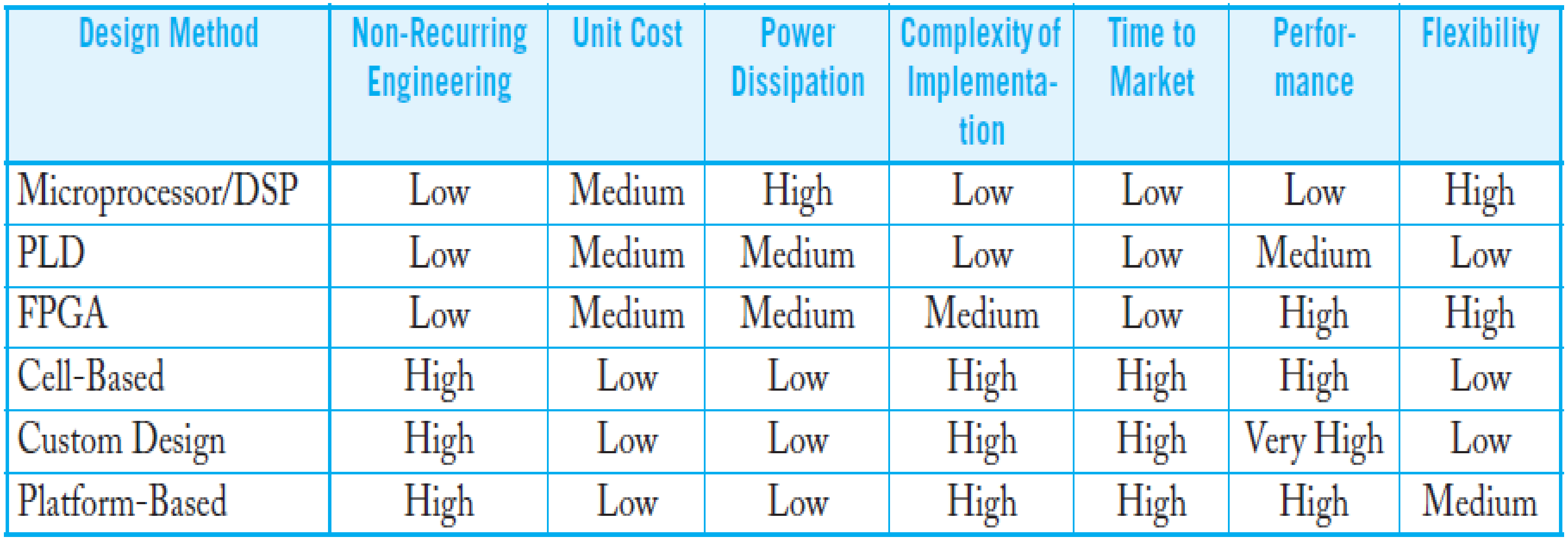


TABLE 14.1 Structure software and VLSI hardware design

Design Principle	Software	Hardware
Hierarchy	Subroutines, libraries	Modules
Regularity	Iteration, code sharing, object-oriented procedures	Datapaths, module reuse, regular arrays, gate arrays, standard cells
Modularity	Well-defined subroutine interfaces	Well-defined module interfaces, timing and loading data for modules, registered inputs and outputs
Locality	Local scoping, no global variables	Local connections through floorplanning

Design Methods

- The base design methods are arranged roughly in order of “increased investment,” which loosely relates to the time and cost it takes to design and implement the system.
- It is important to understand the costs, capabilities, and limitations of a given implementation technology to select the right solution.



Design Method	Non-Recurring Engineering	Unit Cost	Power Dissipation	Complexity of Implementation	Time to Market	Performance	Flexibility
Microprocessor/DSP	Low	Medium	High	Low	Low	Low	High
PLD	Low	Medium	Medium	Low	Low	Medium	Low
FPGA	Low	Medium	Medium	Medium	Low	High	High
Cell-Based	High	Low	Low	High	High	High	Low
Custom Design	High	Low	Low	High	High	Very High	Low
Platform-Based	High	Low	Low	High	High	High	Medium

14.3.1 Microprocessor/DSP

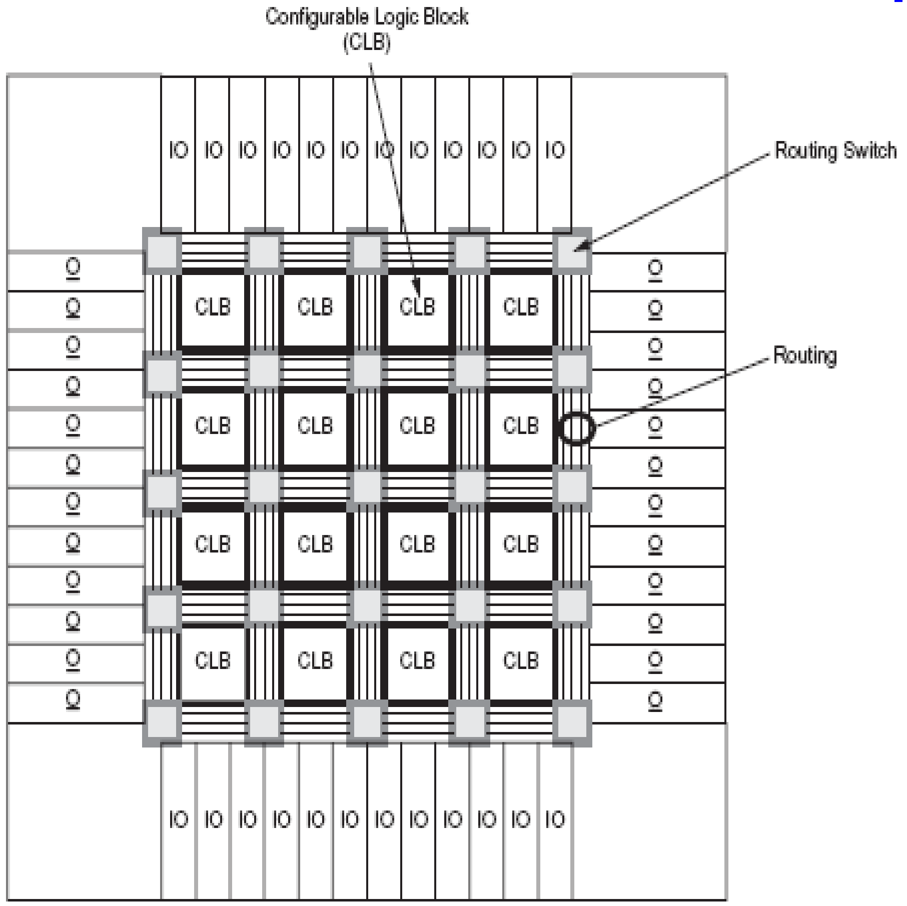
- **Microprocessor/DSP** The practical method to solve a system design problem: use microprocessor or digital signal processor (DSP).
  - Example: PIC family
  - Examples of embedded commercial processor cores include ARM, MIPS, and IBM's PowerPC.

14.3.2 Programmable Logic

- **Programmable Logic:** a variety of programmable chips are efficient than MPs yet faster to develop than dedicated chips:
  - Chips with programmable logic arrays
  - Chips with programmable interconnect
  - Chips with reprogrammable logic and interconnect
- The designer should be familiar with these options for two reasons:
  - Allows the designer to competently assess a particular system requirement for an IC and recommend a solution.
  - Familiarizes the IC designer with methods of making any chip reprogrammable.

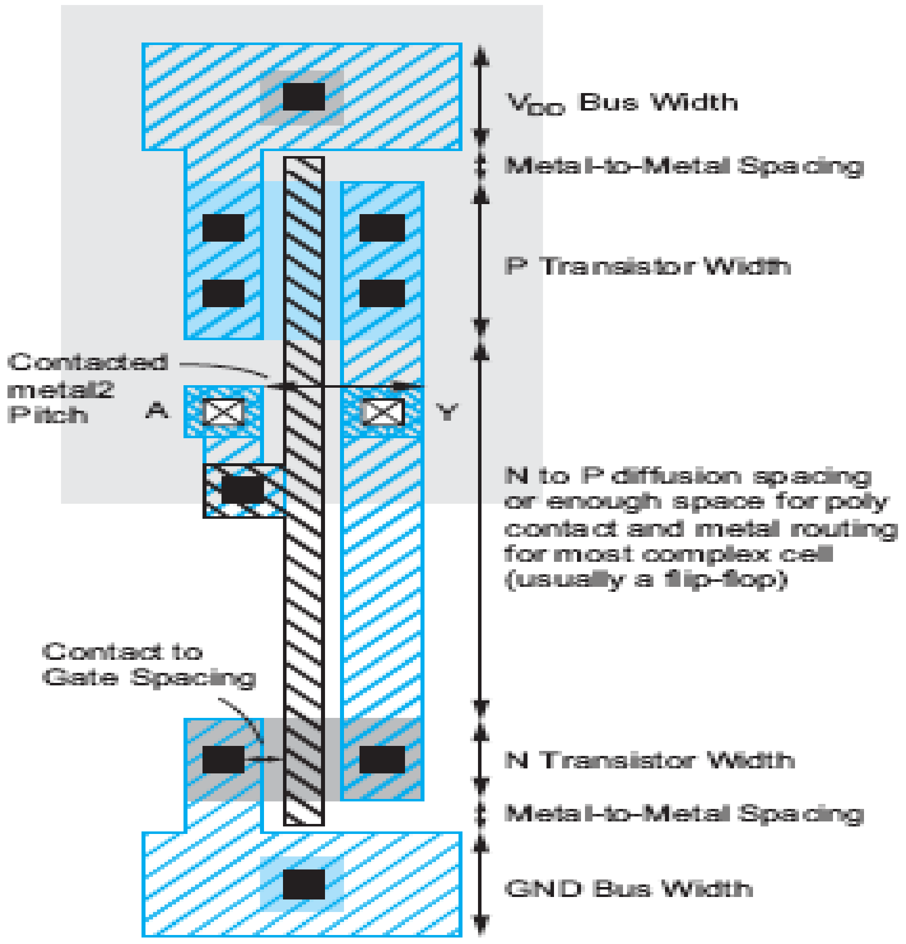
14.3.2 Programmable Logic

- **Programmable Logic Devices:** The devices are descended from chips that implement two-level sum-of-product (AND plane and OR plane to compute any function) programmable logic arrays (PLAs).
- **Field-Programmable Gate Arrays (FPGAs):** use the high circuit densities in modern processes to construct ICs that are completely programmable.



14.3.2 Programmable Logic

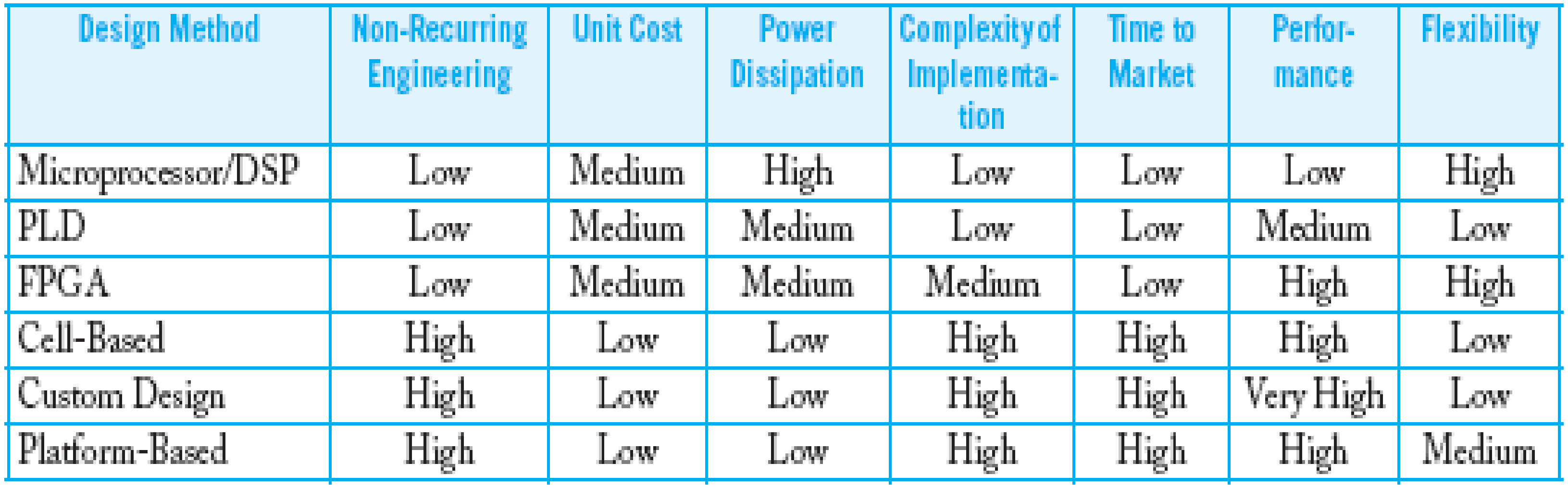
- **Cell-Based Design:** uses a standard cell library as the basic building blocks of a chip. Cells are placed in appropriate positions, then their interconnections are routed. It can deliver smaller, faster, and lower-power chips than FPGAs.



14.3.2 Programmable Logic

- ❑ **Full Custom Design**
- ❑ **Platform-based Design- A System on a Chip:** Implement a design by using common structures such as busses and common high-level languages (such as C) to program the processors.

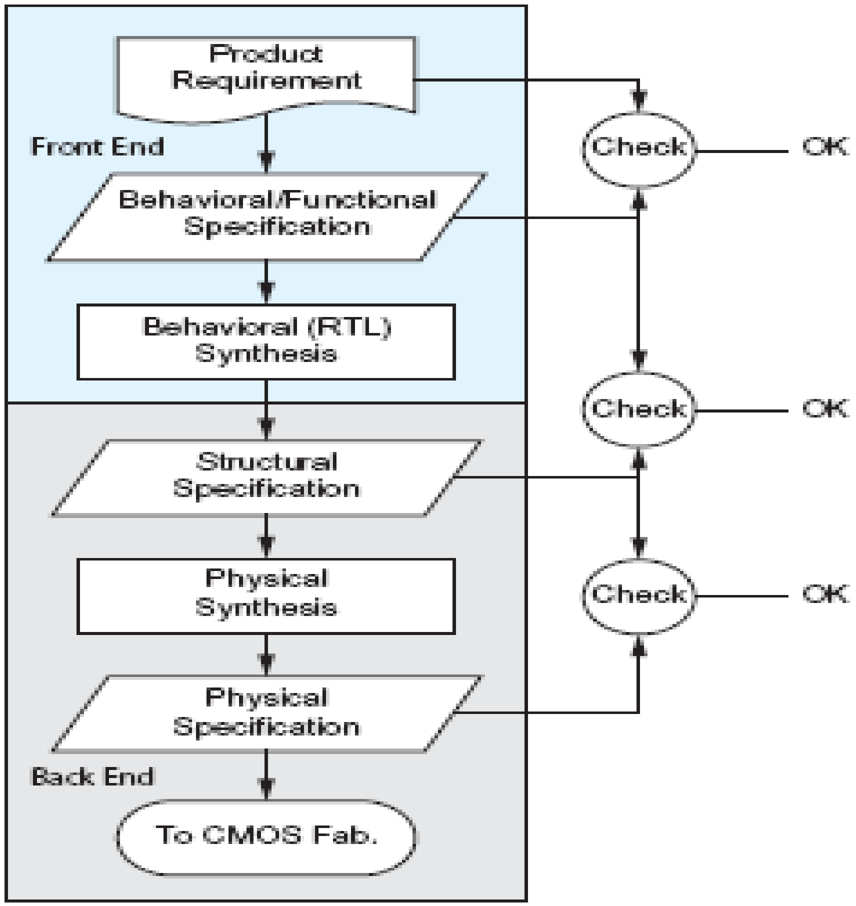
TABLE 14.4 Comparison of CMOS design methods



Design Method	Non-Recurring Engineering	Unit Cost	Power Dissipation	Complexity of Implementation	Time to Market	Performance	Flexibility
Microprocessor/DSP	Low	Medium	High	Low	Low	Low	High
PLD	Low	Medium	Medium	Low	Low	Medium	Low
FPGA	Low	Medium	Medium	Medium	Low	High	High
Cell-Based	High	Low	Low	High	High	High	Low
Custom Design	High	Low	Low	High	High	Very High	Low
Platform-Based	High	Low	Low	High	High	High	Medium

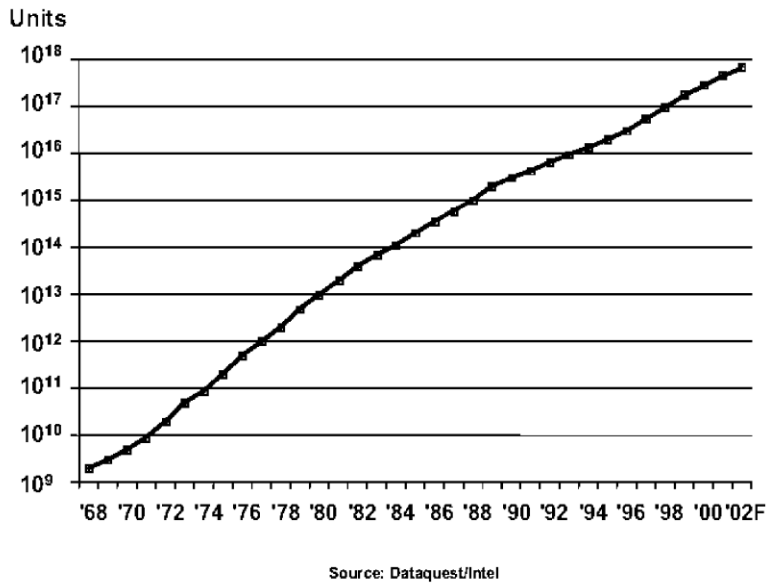
14.4 Design Flow

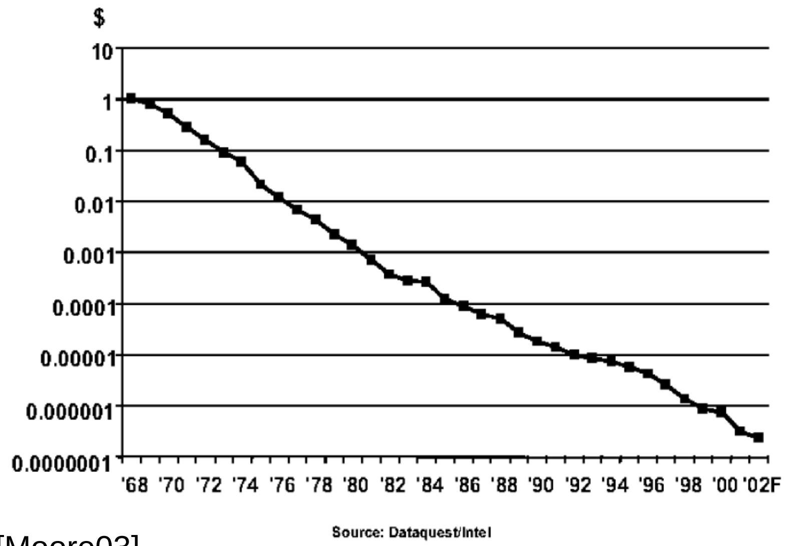
- **Design flow** is a set of procedures that allows designers to progress from a specification for a chip to the final chip implementation in an error-free way.



14.5 Design Economics

- It is important for the IC designer to be able to predict the cost and the time to design a particular IC or sets of ICs. This can guide the choice of an implementation strategy.
- In 2003, $0.01 bought you 100,000 transistors
  - Moore’s Law is still going strong





[Moore03]

Physical Limits

- ❑ Will Moore's Law run out of steam?
  - – Can't build transistors smaller than an atom…
- ❑ Many reasons have been predicted for end of scaling
  - – Dynamic power
  - – Subthreshold leakage, tunneling
  - – Short channel effects
  - – Fabrication costs
  - – Electromigration
  - – Interconnect delay
- ❑ Rumors of demise have been exaggerated

VLSI Economics

- Selling price S_{total}
  - $S_{total} = C_{total} / (1-m)$
- m = profit margin
- C_{total} = total cost
  - Nonrecurring engineering cost (NRE)
  - Recurring cost
  - Fixed cost

14.5.1 NRE

- ❑ Engineering cost
  - – Depends on size of design team
  - – Include benefits, training, computers
  - – CAD tools:
    - Digital front end: $10K
    - Analog front end: $100K
    - Digital back end: $1M
- ❑ Prototype manufacturing
  - – Mask costs: $5M in 45 nm process
  - – Test fixture and package tooling

14.5.2 Recurring Costs

- Once the cost of an IC has been determined, the IC manufacturer will arrive at a price for the specific IC.
- the cost to fabricate an IC is as follows:

$$R_{total} = R_{process} + R_{package} + R_{test}$$

where

- $R_{package}$ = *package cost*
- R_{test} = *test cost—the cost to test an IC is usually proportional to number of vectors* and time to test.
- $R_{process} = W/(N \times Y_w \times Y_{pa})$

where

14.5.2 Recurring Costs

- *W = wafer cost ($500–$5000 depending on process and wafer size)*
- *N = gross die per wafer (the number of complete die on a wafer)*
- *Y_W = die yield per wafer (should be ~70–90+% for moderate-sized dice in a mature* process)
- *Y_{pa} = packaging yield (should be ~95–99%)*
- Note that:
  - Wafer cost / (Dice per wafer * Yield)
  - Wafer cost: $500 - $3000

14.5.2 Recurring Costs

– Dice per wafer: $N = \pi \left[\frac{r^2}{A} - \frac{2r}{\sqrt{2A}} \right]$

Where die area *A and is fabricated on a wafer with radius r.*

– Yield: $Y = e^{-AD}$

- *A is* the size of the die *and D is the average number of defects per unit area.*
- For small A, Y $\approx$ 1, cost proportional to area
- For large A, Y $\rightarrow$ 0, cost increases exponentially

14.5.3 Fixed Costs

- Once a chip has been designed and put into manufacture, the cost to support that chip from an engineering viewpoint may have a few sources:
  - Data sheets
  - Application notes
  - Marketing and advertising
  - Yield analysis

Example

- You want to start a company to build a wireless communications chip. How much venture capital must you raise?

- Because you are smarter than everyone else, you can get away with a small team in just two years:
  - Seven digital designers
  - Three analog designers
  - Five support personnel

Solution

- Digital designers:
  - $70k salary
  - $30k overhead
  - $10k computer
  - $10k CAD tools
  - Total: $120k * 7 = $840k
- Analog designers
  - $100k salary
  - $30k overhead
  - $10k computer
  - $100k CAD tools
  - Total: $240k * 3 = $720k
- Support staff
  - $45k salary
  - $20k overhead
  - $5k computer
  - Total: $70k * 5 = $350k
- Fabrication
  - Back-end tools: $1M
  - Masks: $5M
  - Total: $6M / year
- Summary
  - 2 years @ $7.91M / year
  - $16M design & prototype