Modern VLSI Design — Chapter 1: Overview, Trends, and Design Flow

EECS 168 • Big-picture context, abstraction ladder, IP methodology, and scaling realities.

0) Map of the terrain (what Chapter 1 is really doing)

- **Why VLSI**: Integration brings speed, power, and cost wins at product scale.
- **The scaling story**: Moore's Law history, the end of classic Dennard scaling, and the *power wall*.
- **Design at scale**: Abstraction levels, domains, and the modern VLSI flow.
- **IP as leverage**: Hard vs. soft IP, lifecycle, and practical selection/qualification.
- **Reliability & reality**: Nanometer reliability issues and design-for-yield/manufacturing themes.

Key Idea

VLSI is a two-front war: *technology scaling* (physics, process) and *design scaling* (abstraction, hierarchy, IP). You need both to ship chips on time.

1) Why VLSI matters (integration economics & applications)

Integration wins:

- **Speed**: lower on-chip interconnect parasitics vs. board-level wiring.
- **Energy**: shorter wires + fewer I/O transitions reduce power.
- **Size & cost**: fewer parts, higher reliability, better manufacturability.

Where VLSI shows up: CPUs/MCUs, memory (SRAM/DRAM), I/O/PHY, and accelerators (DSP, vision, ML). Systems are full SoCs, not just processors. *Custom silicon* is widely used across phones, datacenter, networking, and edge devices. [Slides: overview of applications and industry trend] (see sources at end)

2) A quick historical arc (just enough context)

From early mechanical computation → vacuum tubes → the transistor (1947–48) → integrated circuits → microprocessors (Intel 4004, 1971) → deep sub-micron SoCs. Frequency ramped hard through the 90s/early 2000s before thermal and energy walls redirected progress toward parallelism and specialization. [Slides: history snapshots, frequency/power charts]

Gotcha

Raw frequency scaling lost steam; the modern playbook is *parallelism* (multicore), *specialization* (accelerators), and *system design* (memory, packaging, interconnects).

3) Scaling realities: Moore, Dennard, and the power wall

3.1 Moore's Law (transistor count)

For decades, transistors/chip doubled every ∼18–24 months, enabling more cores, cache, and features. [Slide index shows Moore's Law content]

3.2 Frequency and power density

Lead microprocessor **frequency** rose rapidly, but **power** and **power density** rose too; delivering and dissipating that power became prohibitive. [Frequency/Power/Power-density charts]

Key Idea

Power wall: Frequency stalled around a few GHz due to thermal limits; pushing V or f further exploded power density, overwhelming cooling budgets.

3.3 Dennard scaling ended

Classic Dennard: as devices shrink, power density stays constant (voltage scales with length). That assumption broke; leakage rose, voltage scaling slowed, power density soared. [Slides: "End of Moore/Dennard/Power Wall"]

Checklist

Implications:

- Energy efficiency is king: performance/ W matters more than raw f.
- Architecture must carry more: multicore, data-parallel units, accelerators.
- Packaging/interconnect innovations (chiplets, advanced 2.5D/3D) join the critical path.

Gotcha

"Just crank the clock" is done. Gains now come from *doing less work* per result (specialization) or *doing work in parallel* at lower V/f.

4) The VLSI design problem (why we need abstraction)

Conflicting objectives: area, performance, energy/power, reliability, testability, and time-to-market clash. Designs span *multiple abstraction levels*, from transistors up to CPUs. [Slides: "Challenges in VLSI design"]

Key Idea

Divide–and–conquer: limit what a human must reason about at once. Use hierarchy (compose from verified blocks) and abstraction (only the right detail per level).

5) Abstraction levels and design domains (the ladder)

5.1 Abstraction ladder

- **Specification**: English/system models, performance and cost goals.

- **Behavioral**: executable spec/high-level synthesis targets.
- **Register-Transfer (RTL)**: state machines, datapaths, cycle-level timing.
- **Logic**: gate-level structures, Boolean transforms, basic timing arcs.
- **Circuit**: transistor-level choices, device sizing, analog effects.
- **Layout**: geometric shapes, layers, design rules, exact parasitics.

Each level has its own modeling objects and metrics (e.g., throughput vs. nanoseconds vs. microns). [Slides: design abstractions levels]

5.2 Domains & methodologies

Top-down refines specs to masks; bottom-up builds reusable components. Real flows interleave both (e.g., pick a verified memory macro, then architect around it). [Slides: design domains (Top-down/Bottom-up)]

Checklist

At each level know: key objects (e.g., FSM, netlist, layout), validation method (simulation, STA, DRC/LVS), and handoff artifacts (constraints, libs, LEF/DEF, GDS).

6) The end-to-end VLSI flow (what happens when)

1. **Specification**: function, PPA targets, interfaces, cost constraints.
2. **Architecture**: partition into blocks; choose memory hierarchies, datapaths; estimate PPA.
3. **Logic/RTL**: write/verify RTL, define protocols, pipeline for timing.
4. **Circuit**: size devices, select standard cells/Vt options; clock tree and IO planning begin.
5. **Physical (layout)**: floorplan, place, route, extract parasitics, close timing/power/SI.

Design validation happens continuously: forward check between levels; back-annotate real timing/power upstream once parasitics land. **Manufacturing test** (after fab) screens for defects and speed-grades parts; it is *not* the same as design validation. [Slides: design process; design validation; manufacturing test]

Key Idea

Fixing a bug gets exponentially more expensive the later you find it. Validate early, validate often.

7) IP-based design (how big chips ship on time)

7.1 Types of IP

Hard IP: finished layout (fixed pins/layers); process-qualified; great for tight PPA and analog/PHYs. Must meet layout pin placement, layer usage, and transistor sizing standards.

Soft IP: RTL/gate netlist; you synthesize/P&R; timing is only firm after layout; wrap to meet interface standards. [Slides: IP, hard vs. soft IP]

7.2 IP across the hierarchy and the lifecycle

Across hierarchy: standard cells, memories, RTL blocks, CPUs, buses, I/O devices.

Lifecycle: spec → HDL design → validation & documentation/extraction → qualification → IP database/modules → chip integration.

Using IP: identify candidates (vendor, open, internal), evaluate suitability, license as needed, and consider qualification if pushing analog limits. [Slides: IP across hierarchy; IP lifecycle; Using IP]

Checklist

When choosing IP: hard/soft, functionality, PVT corners, power, DRC/LVS readiness, interface compliance, deliverables (models, libs, views), support policy.

8) Reliability and yield (design for the real world)

Nanometer techs bring EM (electromigration), SM, NBTI, hot carriers, TDDB concerns. DFM/DFY practices and architecture/circuit techniques can compensate for weaker devices or variability to improve yield. [Slides: Reliability; DFM/DFY]

Gotcha

Running closer to physical limits without reliability guardrails saves area today and costs you *lifetime* and *RMA* tomorrow.

9) Managing complexity (how we keep our sanity)

Divide–and–conquer: group small components into larger modules; reuse proven blocks; maintain clean interfaces. [Slides: Dealing with complexity]

Hierarchical names & netlists: instances are typed (e.g., many adders of one type). Hierarchical names (`top/u_core/u_alu.sum`) disambiguate signals across levels. Netlists connect pins to nets; component lists bind instance types. [Slides: hierarchical names, component hierarchy, netlists]

Key Idea

Good hierarchy and naming aren't bureaucracy—they're how timing closure and debugging remain tractable on million-gate designs.

10) Power: the quick working model (for context)

Dynamic $P_{\text{dyn}} \approx \alpha C V^2 f$ (activity α, switched capacitance C, supply V, clock f).

Static $P_{\text{leak}} \approx I_{\text{leak}} V$ (dominant in advanced nodes).

Why it matters here: Chapter 1's frequency/power/power-density charts motivate all later emphasis on low-power architecture and physical design choices.

Checklist

Levers: reduce V, reduce C, reduce f, reduce activity α, power-gate blocks, use higher-Vt cells where timing allows, and move compute where energy is cheapest.

11) What to be able to do after this chapter

- Walk the abstraction ladder and name artifacts/metrics for each level.
- Explain Moore vs. Dennard vs. the power wall in one minute.
- Describe the VLSI flow end-to-end and where validation vs. manufacturing test fits.
- Distinguish hard vs. soft IP; outline an IP evaluation checklist.
- Name the major reliability risks at nanometer nodes and why DFY/DFM exists.

Appendix: Mini-Worksheet (practice, keep unsolved)

A) Abstractions drill

For each level (spec, behavioral, RTL, logic, circuit, layout):
(i) name the primary **modeling object**, *(ii)* one **metric**, *(iii)* one **validation method**, *(iv)* the **handoff artifact** to the next stage.

B) IP decision exercise

You need a PCIe Gen4 x8 controller. Draft a 7-point checklist comparing a vendor hard IP vs. an open soft IP for your node and PPA targets. Include: deliverables, PVT coverage, interface wrappers, timing views, qual data, support, and licensing.

C) Scaling story in 6 sentences

Write six sentences that connect: Moore's Law → frequency rise → power density → power wall → multicore/accelerators → packaging/interconnect innovation.

D) Reliability scan

Pick two risks (e.g., EM, NBTI). State what physical stress causes them and one circuit/architecture mitigation for each.

End of expanded notes — bring this to class; fill the worksheet items during lecture.