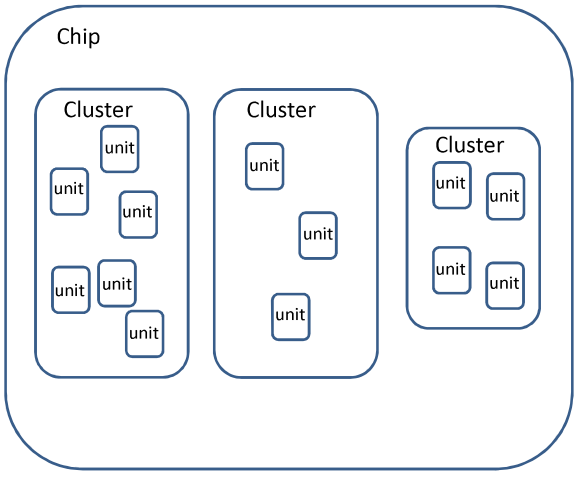
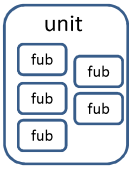
Product lines and project lifecycles
Design Validation

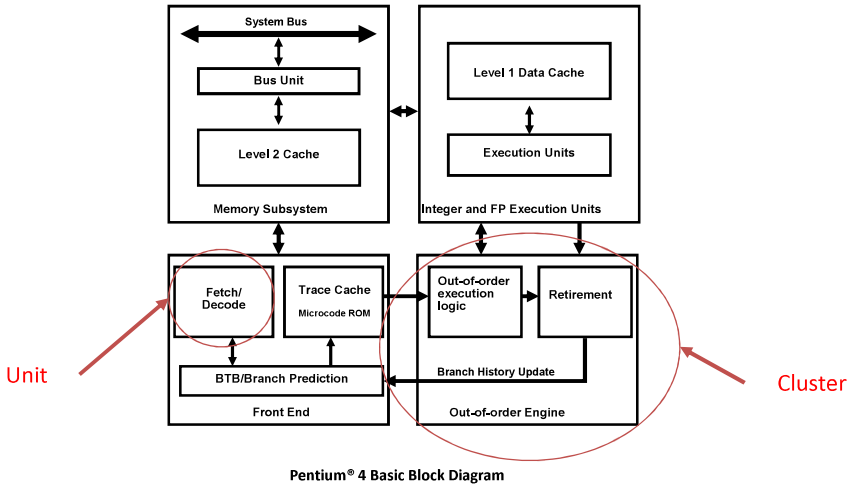
Nomenclature





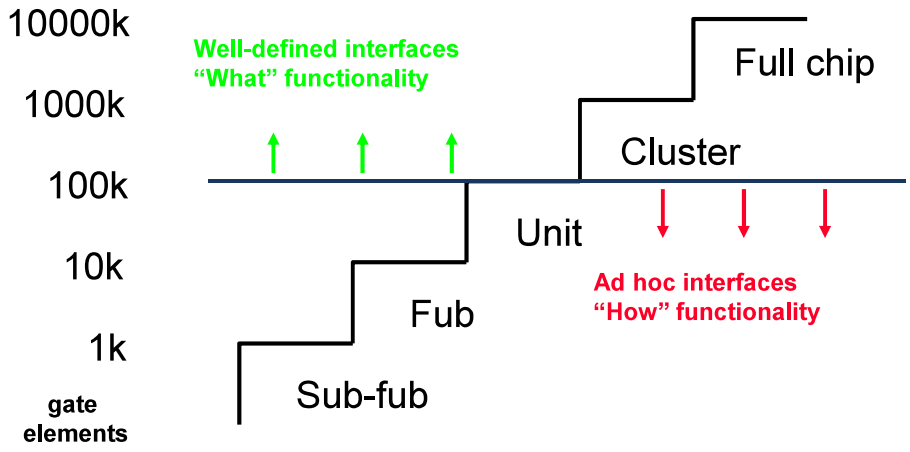
Hierarchy:
- Chip
- Core/processing units
- Cluster
- Unit
- Fub

Design Hierarchy

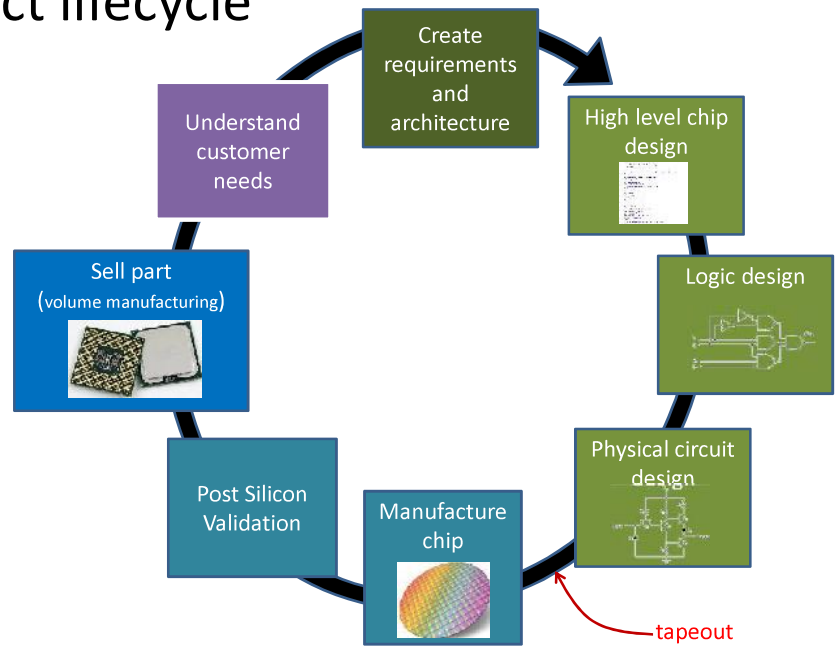


Pentium® 4 Basic Block Diagram

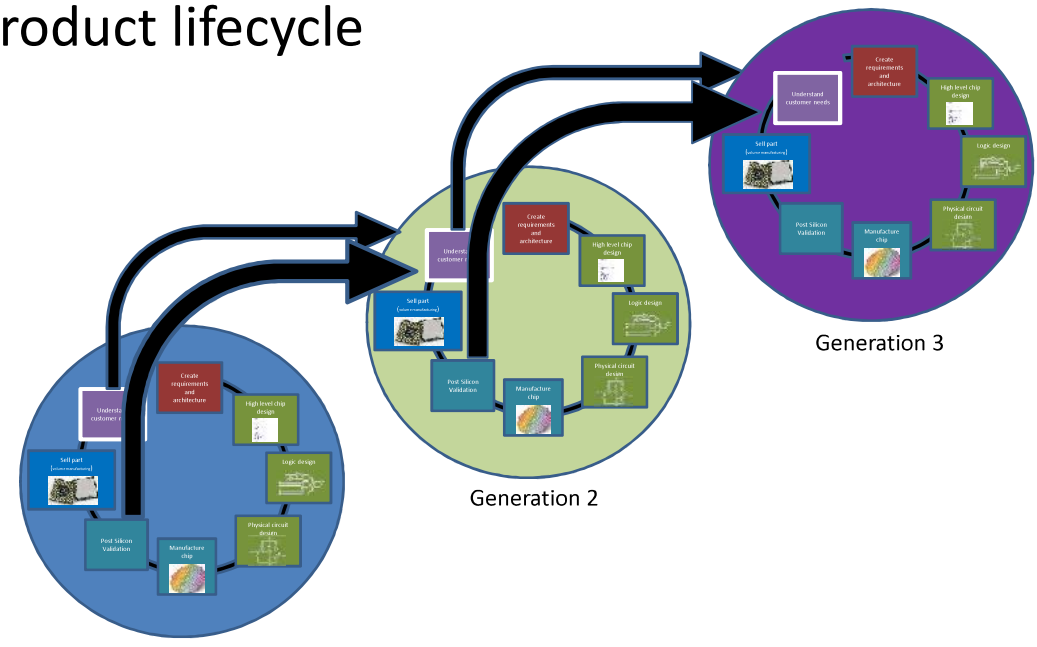
Design Hierarchy



Product lifecycle

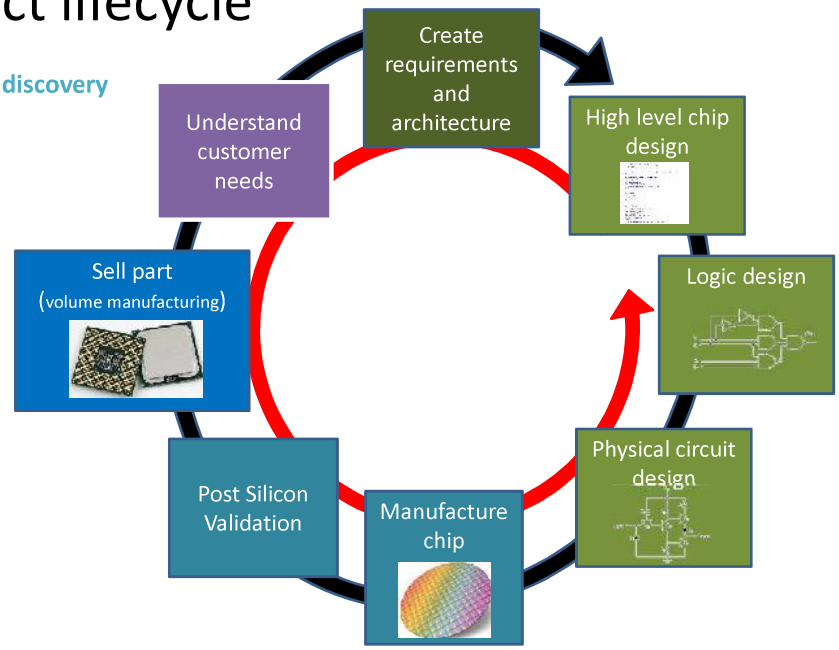


Product lifecycle

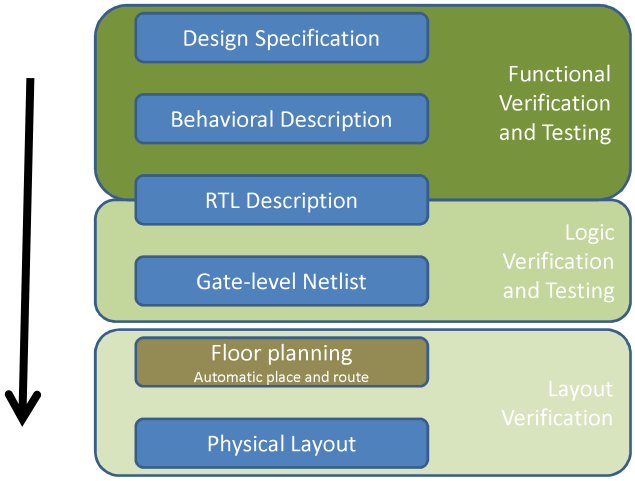


Product lifecycle

Cost of late discovery

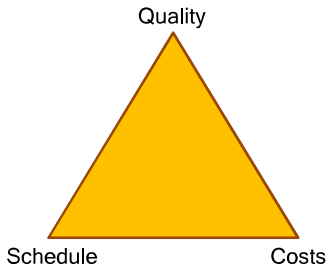


Pre-Silicon Design Flow



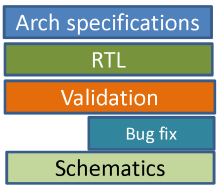
Quality-Schedule-Costs

- Design development environment
  - Deliverables
    - Specifications, RTL, validation
  - Schedule
    - Pre-Silicon goal to get to healthly tapeout
    - Provide
  - Costs
    - Equipment, people, training, tools,
    - (wasted) time

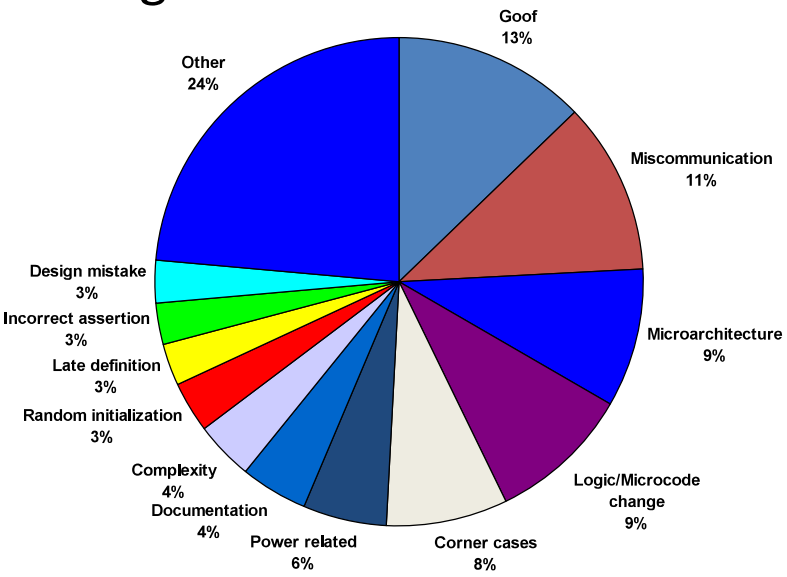


➔ Thoughtful planning and monitoring throughout the process

Schedule



Where Do Bugs Come From?



Intel Pentium™ 4 Pre-silicon Bug Breakdown (Bentley HLDVT'02)

Role of Design Validation

- Product quality gatekeeper (design organization task)
  - Prevent microarchitecture and design bugs
  - Identify bugs and determine fixes for bugs
  - Repeat on multiple models and generations
  - Assess the health of a design
  - Move the project forward

- What makes a good validator?
  - Deep understanding of architecture/microarchitecture
  - Excellent problem solving skills
  - Expert on validation methodology and tools
  - Able to work with ambiguity
  - Disciplined
  - An active knowledge transfer agent
  - Continuously improve

http://www.kgw.com/sports/US-Goalie-Tim-Howard-is-saving-EVERYTHING-265698771.html

When is validation done?

- Never
- Space too large
- Spec ambiguous and very rarely incomplete.
- How many bugs found, but not how many in the design
  - Guidelines from past experience
- How much validation?
  - Exercise
  - Bug hunting validation, coverage driven validation
  - Robust design validation
- Validation cost models
  - Cost of escapee overtime (in $ and time)
  - Hardware and Software costs different

Bob Bentley's 10 Validation Commandments

- A bug prevented is a bug that you don't have to find
- Finding bugs is good
- Not all bugs are equally important
- If it isn't tested, it doesn't work
- If it doesn't have to work, we could ship it tomorrow
- A test is only a means to an end, not an end in itself
- Coverage is only a means to an end, not an end in itself
- Overlap is good
- Surviving bugs are Darwinian
- Testing only shows the presence of bugs, not their absence

Applying Sun Tzu's "on the art of war" to verification

- Sun Tzu's 13 chapters
  1. Laying Plans
  2. Waging War
  3. Attack by Stratagem
  4. Tactical Dispositons
  5. Energy
  6. Weak Points & Strong
  7. Maneuvering
  8. Variation in Tactics
  9. The Army on the March
  10. Terrain
  11. The Nine Situations
  12. The Attack by Fire
  13. The Use of Spies

Brian Bailey, 2008, "Is it time to declare a verification war?"

"Bring Good Things to Life"

- New user experiences
- Features, architectures, microarchitectures, designs.

Engineering process

- How do we organize and coordinate design development activities?
  - fast, efficient
- Measure progress
- How do we improve over time?
- Teamwork

Specification and Verification Theory

- Defining what designs do
  - Languages
  - Translation/synthesis to physical item
- Clear and concise specifications
- Reasoning about specifications
  - Mathematical systems that can predict real behavior
- Large complex, concurrent systems