Lecture 3: Design Methodologies

Embedded Computing Systems

Mikko Lipasti, adapted from M. Schulte

Based on slides and textbook from Wayne Wolf

Topics

- Design Goals
- Design methodologies.
- Methodologies and standards.

Design goals

- Functional requirements: input/output relations.
- Non-functional requirements: cost, performance, power, etc.
- Some project goals may be difficult to measure.
  - What types of goals are more difficult to measure?
  - Why are these goals important?

Aspects of performance

- Embedded system performance can be measured in many ways:
  - Average vs. worst-case vs. best-case.
  - Throughput vs. latency.
  - Peak vs. sustained.
- Why might we care about best-case performance? Average-case? Worst-case?
- How is performance estimated/measured?

Energy/power

- Energy consumption (joules) is important for battery life.
- Power consumption (Watts = joules/sec) is important for heat generation or for generator-powered systems (e.g. cars).
- What are some techniques for improving energy and power consumption?

Cost

- *Manufacturing costs*
  - Determined by the cost of components and the manufacturing process used
  - Must be paid off across all the systems.
    - Hardest in small-volume applications.
  - Incurred for each device
- *Designed costs* determined by labor and the equipment used to support the design process
- *Lifetime costs* include software and hardware maintenance and upgrades.

Other design attributes

- Design time must be reasonable. May need to finish by a certain date.
  - Time to market
- System must be reliable; reliability requirements differ widely.
- Quality includes reliability and other aspects: usability, durability, etc.
- What other attributes may be important in embedded systems?

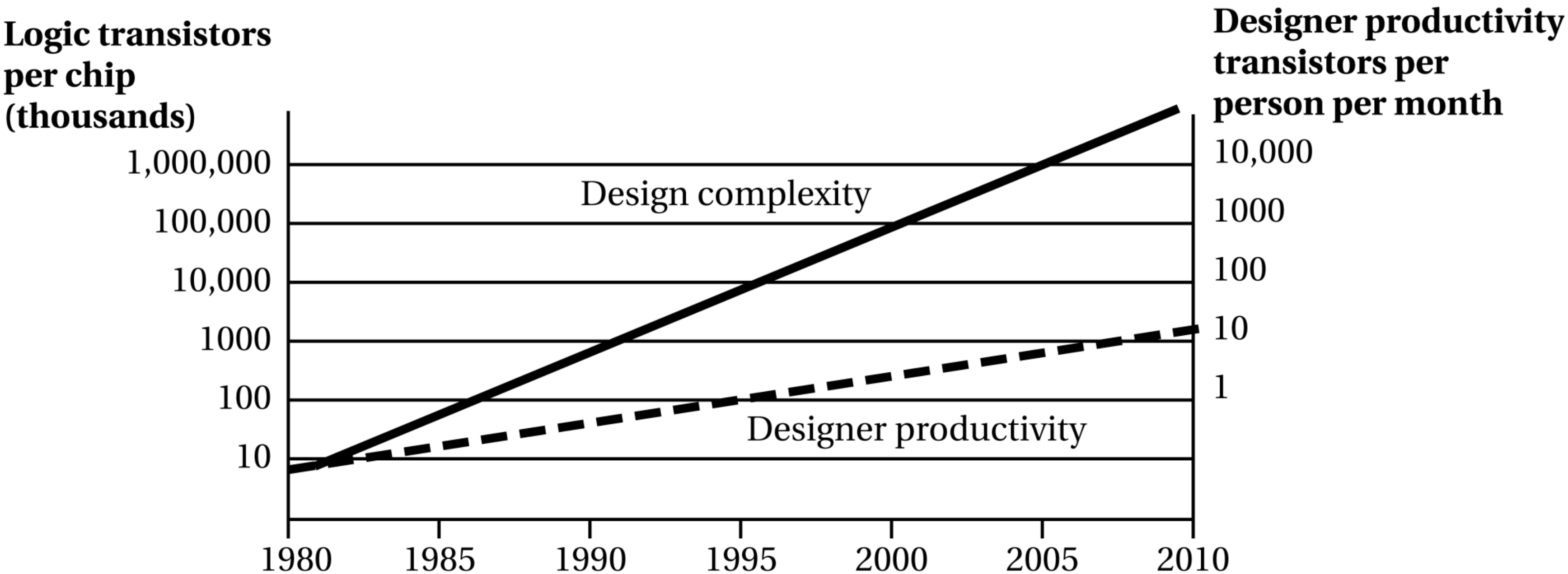
Design methodology

- Design methodology: a procedure for creating an implementation from a set of requirements.
- Methodology is important in embedded computing:
  - Must design many different systems.
  - We may use same/similar components in many different designs.
  - Both design time and results must be predictable.

Embedded system design challenges

- Design space is large and irregular.
- We don’t have synthesis tools for many steps.
- Can’t simulate everything.
- May need to build special-purpose simulators quickly.
- Often need to start software development before hardware is finished.

Design complexity vs. designer productivity

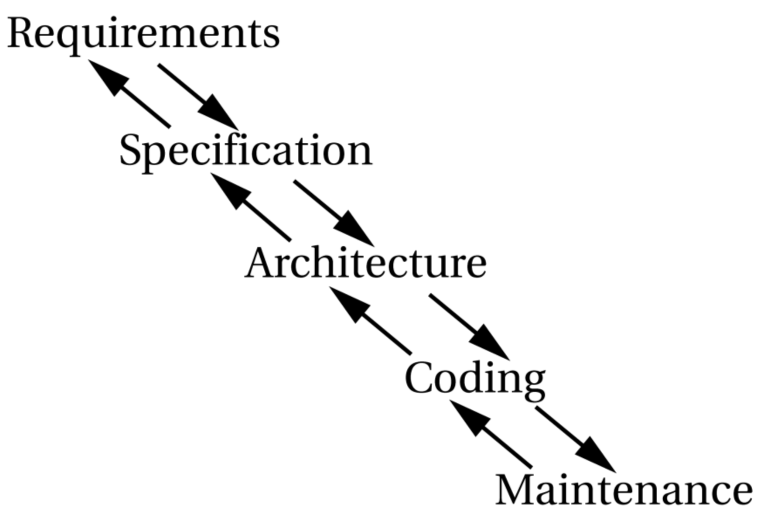


- How can design complexity be managed?
- How can designer productivity be improved?

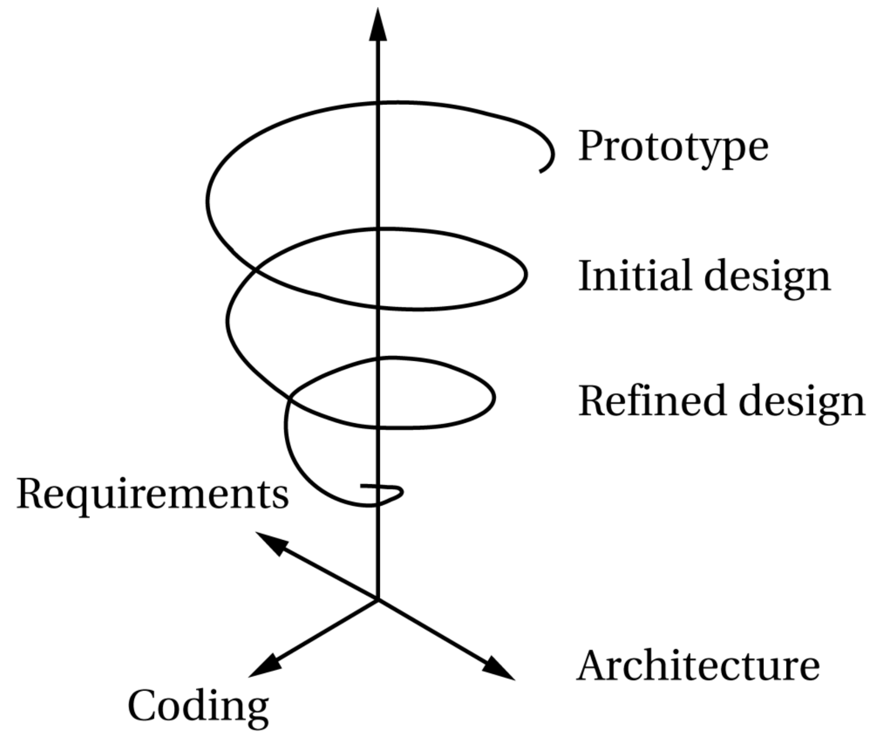
Basic design methodologies

- Figure out flow of decision-making.
- Determine when bottom-up information is generated.
- Determine when top-down decisions are made.

Software design methodologies: waterfall and spiral models

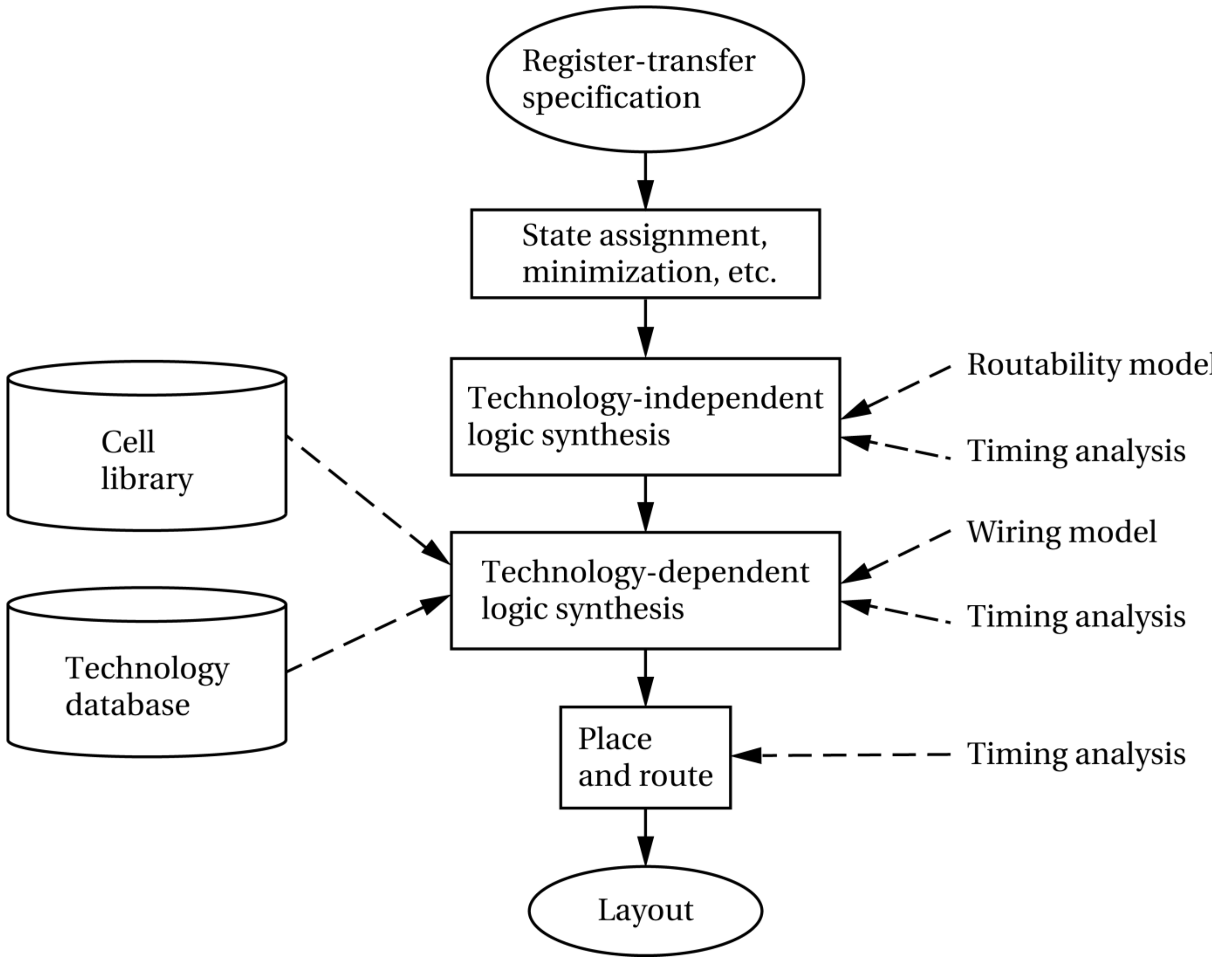


Waterfall

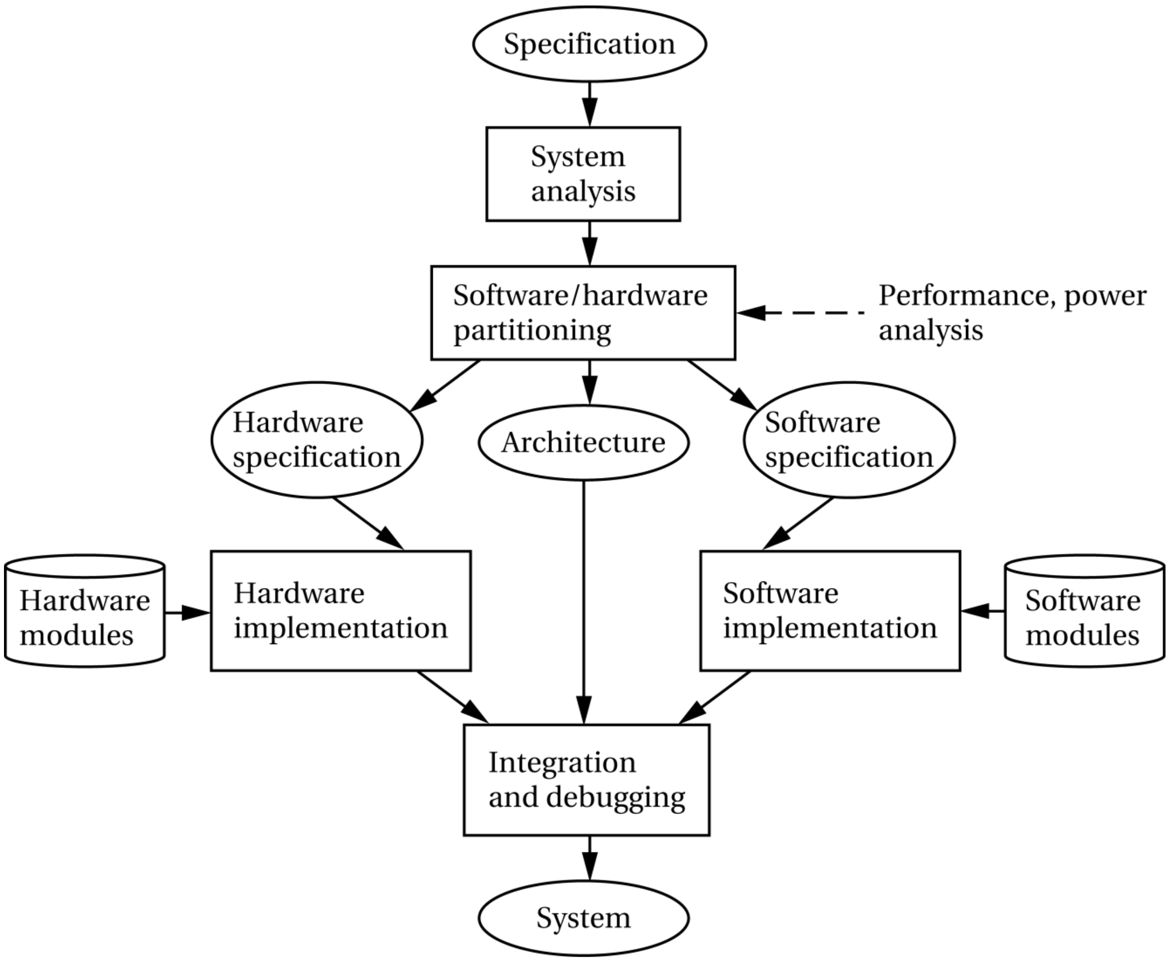


Spiral

Hardware design flow

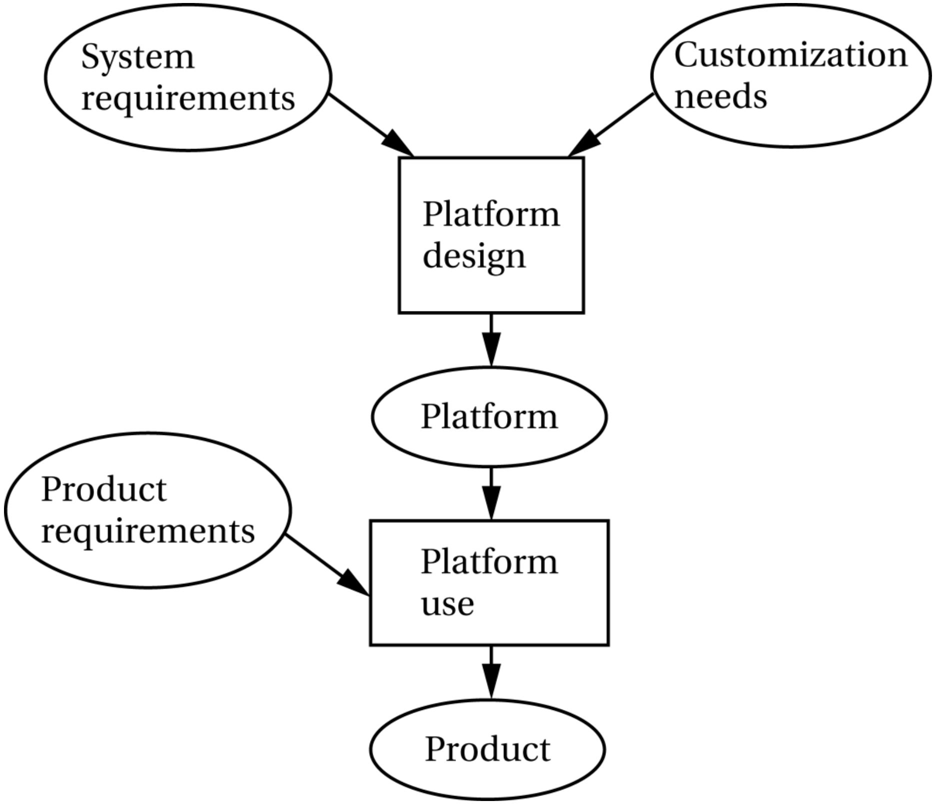


Hardware/software co-design flow



Platform-based design

- Platform includes hardware, supporting software.
- Two stage process:
  - Design the platform.
  - Use the platform.
- Platform can be reused to host many different systems.



Platform design

- Turn system requirements and software models into detailed requirements.
  - Use profiling and analysis tools to measure existing executable specifications.
- Explore the design space manually or automatically.
- Optimize the system architecture based on the results of simulation and other steps.
- Develop hardware abstraction layers and other software.

Programming platforms

- Programming environment must be customized to the platform:
  - Multiple CPUs.
  - Specialized memory.
  - Specialized I/O devices.
- Libraries are often used to glue together processors on platforms.
- Debugging environments are a particular challenge.

Standards-based design methodologies

- Standards enable large markets.
- Standards generally allow products to be differentiated.
  - Different implementations of operations, so long as I/O behavior is maintained.
  - User interface is often not standardized.
- Standard may dictate certain non-functional requirements (power consumption, latency) and implementation techniques.

Reference implementations

- Executable program that complies with the I/O behavior of the standard.
  - May be written in a variety of languages.
- In some cases, the reference implementation is the most complete description of the standard.
- Reference implementation is often not well-suited to embedded system implementation:
  - Single process.
  - Infinite memory.
  - Non-real-time behavior.

Designing standards-based systems

1. Design and implement system components that are not part of the standard.
2. Perform platform-independent optimizations.
3. Analyze optimized version of reference implementation.
4. Design hardware platform.
5. Optimize system software based on platform.
6. Further optimize platform.
7. Test for conformity to standard.

H/264/AVC

- Implements video coding for a wide range of applications:
  - Broadcast and videoconferencing.
  - Cell phone-sized screens to HDTV.
- Video codec reference implementation contains 120,000 lines of C code.

Design verification and validation

- Testing exercises an implementation by supplying inputs and testing outputs.
- Validation compares the implementation to a specification or requirements.
- Verification may be performed at any design stage; compares design at one level of abstraction to another.

Design verification techniques

- Simulation uses an executable model, relies on inputs.
- Formal methods generate a (possibly specialized) proof.
- Manual methods, such as design reviews, catch design errors informally.

A methodology of methodologies

- Embedded systems include both hardware and software.
  - HW, SW have their own design methodologies.
- Embedded system methodologies control the overall process, HW/SW integration, etc.
  - Must take into account the good and bad points of hardware and software design methodologies used.

Useful methodologies

- Software performance analysis.
- Architectural optimization.
- Hardware/software co-design.
- Network design.
- Software verification.
- Software tool generation.

Joint algorithm and architecture development

- Some algorithm design is necessarily performed before platform design.
- Algorithm development can be informed by platform architecture design.
  - Performance/power/cost trade-offs.
  - Design trends over several generations.

Summary

- Design Goals
- Design methodologies.
- Methodologies and standards.