CprE 488 – Embedded Systems Design

Lecture 1 – Introduction

Phillip Jones
Electrical and Computer Engineering
Iowa State University

www.ece.iastate.edu/~phjones
rcl.ece.iastate.edu

The trouble with computers, of course, is that they're very sophisticated idiots. They do exactly what you tell them at amazing speed – The Doctor

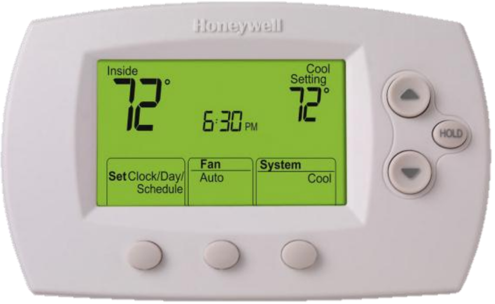
What is an Embedded System? (CPRE 288 reminder)

- Your Definition?
- What are some properties of an Embedded System?

Quadcopter

Micro SD Card?

Blu-Ray / Remote



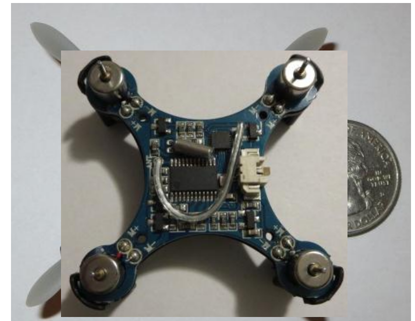
Programmable Thermostat

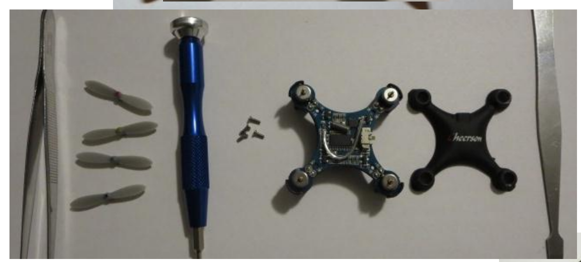
Roomba

What is an Embedded System? (CPRE 288 reminder)

- Your Definition?
- What are some properties of an Embedded System?

Quadcopter





Micro SD Card?

Blu-Ray / Remote

Programmable Thermostat

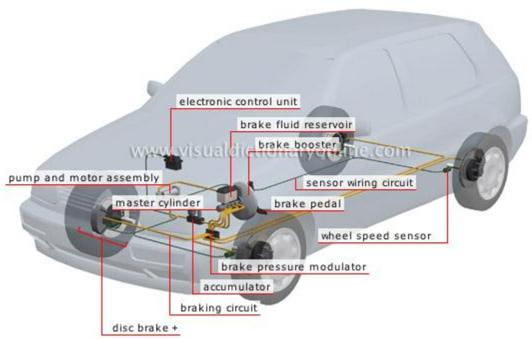
Roomba

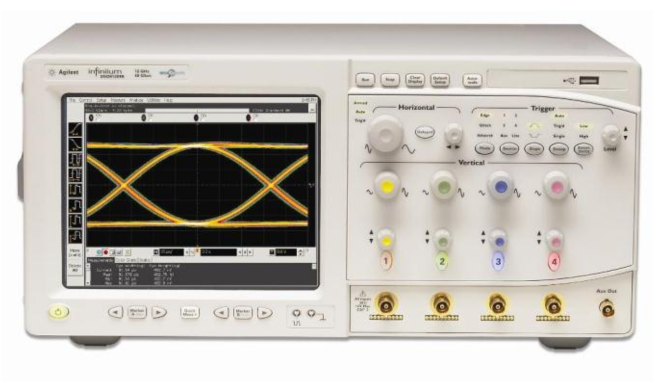
What is an Embedded System?

- The textbook definitions all have their limits
- An **embedded system** is simultaneously:
  1. "a digital system that provides service as part of a larger system" – *G. De Micheli*
  2. "any device that includes a programmable computer but is not itself a general-purpose computer" – *M. Wolf*
  3. "a less visible computer" *- E. Lee*
  4. "a single-functioned, tightly constrained, reactive computing system" – *F. Vahid*
  5. "a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints" – *Wikipedia*

Perspective Matters!

- These definitions quickly become blurred when changing perspective:

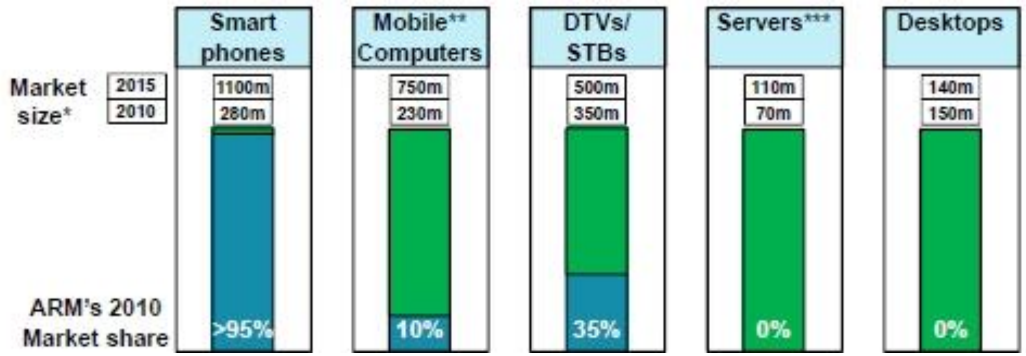


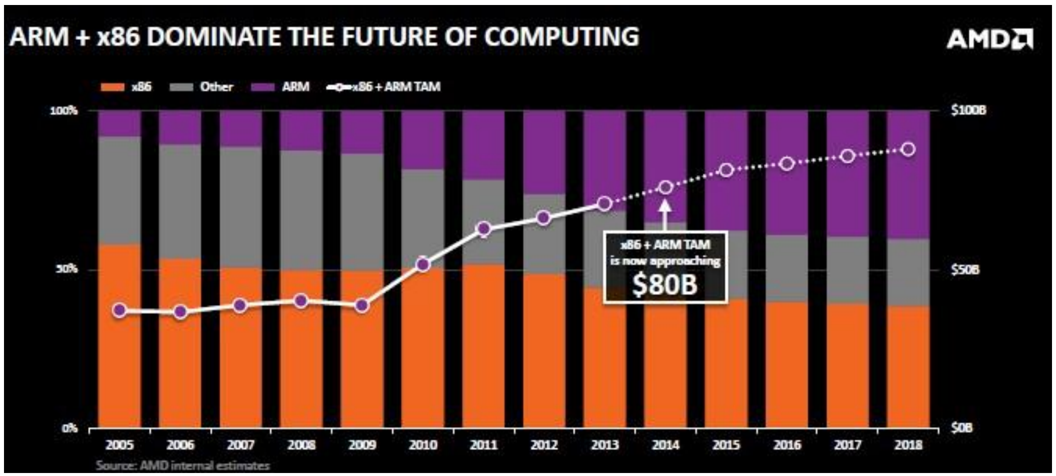


	✓	✗
Part of a larger system:		
Not general-purpose:		
Less visible:		
Tightly constrained:		
Dedicated function:		

Another Practical Definition

- An embedded system is a computing system that uses an ARM processor

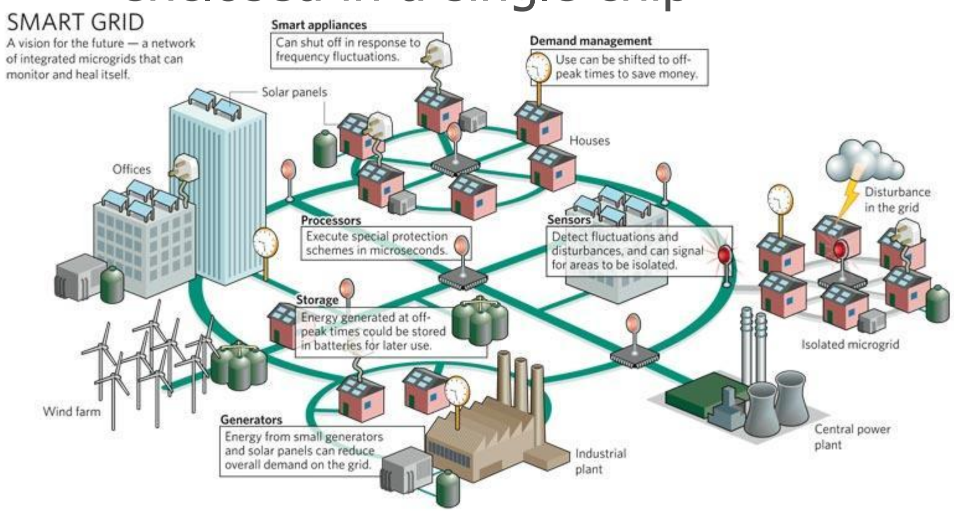


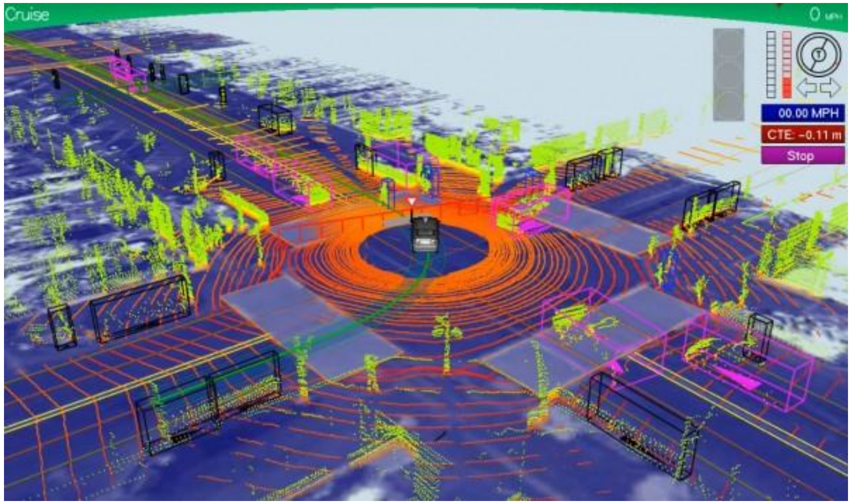


- Multiple caveats:
  - There is a significant 8-bit embedded market as well (e.g. PIC, Atmel, 8051)
  - ARM is also attempting to grow into the desktop and server market

A Different Paradigm

- Cyber-Physical System (CPS): an integration of computation with physical processes
  - Embedded computers monitor and control the physical processes
  - Feedback loops – physical processes affect computation, and vice versa
- Examples tend to include networks of interacting components (as opposed to standalone embedded devices)
  - Still a matter of perspective as networks can span continents or be enclosed in a single chip





Scale of Embedded Devices

- Even as Electrical and Computer Engineers it can be easy to understate the scale (both in terms of size and ubiquity) of embedded devices

- SanDisk microSD card
- 100 MHz ARM CPU

- Apple Lightning Digital AV Adapter
- 256 MB DDR2, ARM SoC

- But for what reason?

This Course's Focus

- *Embedded system design* – the methodologies, tools, and platforms needed to **model**, **implement**, and **analyze** modern embedded systems:
  - Modeling – specifying what the system is supposed to do
  - Implementation – the structured creation of hardware and software components
  - Analysis – understanding why the implementation matches (or fails to match) the model
  - Design is not just hacking things together (which is admittedly also fun)

- What makes embedded system design uniquely challenging?
  - System reliability needs:
    - Can't crash, may not be able to reboot
    - Can't necessarily receive firmware / software updates
  - System performance and power constraints:
    - Real-time issues in many applications
    - (Potentially) limited memory and processing power
  - System cost:
    - Fast time to market on new products
    - Typically very cost competitive

CprE 488 Survival Skills

Necessary Skill	Gained From
Software Development (General)	
Pointers	
Memory and Peripheral Interfacing	
CPU Architecture	
HDL Design	
Circuits and Signals	
Critical Thinking	
Planning and Hard Work	

- Any course that claims to teach you how to design embedded systems is somewhat misleading you, as the technology will continue to undergo rapid change
- Our goal: provide a fundamental understanding of existing design methodology coupled with some significant experience on a current state-of-the-art platform

CprE 488 – Meet the Staff

Prof. Phillip Jones
phjones@iastate.edu
Office Hours: TBA (329 Durham)

Instructor

Robert Wernsman

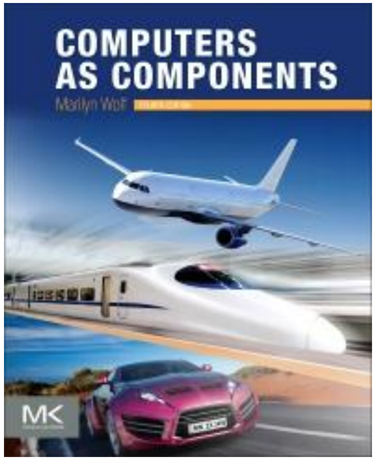
James Talbert

cpre488-tas@iastate.edu
Office Hours: TBA (Lab)

Teaching Assistants

CprE 488 – Resources

- Main text: M. Wolf. *Computers as Components (4th edition): Principles of Embedded Computing System Design*, Morgan Kaufmann, 2017.



- We are here to help, but communication is key!
- Key online resources:
  - Class webpage: class.ece.iastate.edu/cpre488 – contains lecture notes, assignments, documentation, general schedule information (Note: HW0 is due this Friday!!)
  - Canvas space: https://canvas.iastate.edu is heavily used for announcements, discussion, online submission, grading
  - Class wiki: wikis.ece.iastate.edu/cpre488 – updated (by you!) to include general tips and tricks and project photos/videos

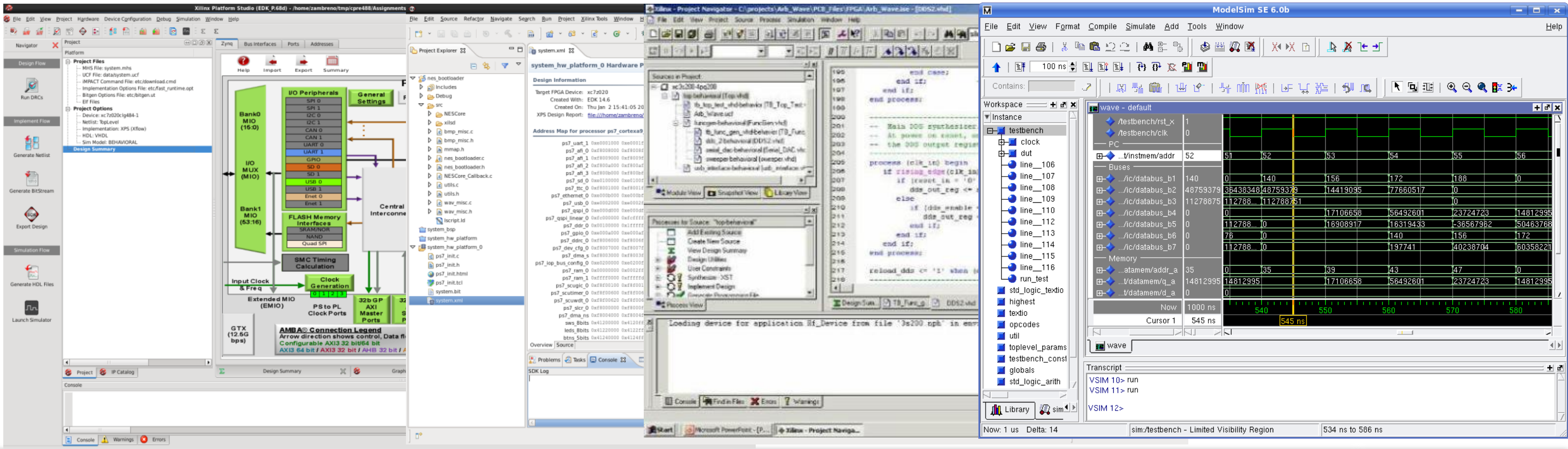
Weekly Layout (Office hours to add)

	Monday	Tuesday	Wednesday	Thursday	Friday
9 am			Lab A (2041 Coover)		Lab C (2041 Coover)
10 00					
11 00					
12 pm					
1 00		Lecture (1126 Sweeney)		Lecture (1126 Sweeney)	
2 00					
3 00					
4 00				Lab B (2041 Coover)	
…					
7 00					

FPGA Design Tools

- Necessary steps (scripts will be provided to help automate):
  1. Login to any of the departmental Linux machines:
     - Remote access to machines on the list here: http://it.engineering.iastate.edu/remote/
     - Use Cygwin/X (ssh), NX client (preferred if the machine supports it)
     - Some extra machines connected to FPGA boards if you *really* need them
  2. From the bash shell, enter the following:

```
source /remote/Xilinx/14.6/settings64.sh
export PATH=$PATH:/remote/Modelsim/10.1c/modeltech/linux_x86_64/
export LM_LICENSE_FILE=1717@io.ece.iastate.edu:27006@io.ece.iastate.edu
```



Lecture Topic Outline

- ✓ Lect-01: Introduction
- Lect-02: Embedded Platforms
- Lect-03: Processors and Memory
- Lect-04: Interfacing Technologies
- Lect-05: Software Optimization
- Lect-06: Accelerator Design
- Lect-07: Embedded Control Systems
- Lect-08: Embedded OS

Machine Problems (MPs)

- 5 team-based, applied assignments
  - Graded on completeness and effort
  - Significant hardware and software components
  - Two weeks each, with in-class and in-lab demos

- Tentative agenda:
  - MP-0: Platform Introduction
  - MP-1: Quad UAV Interfacing
  - MP-2: Digital Camera
  - MP-3: Target Acquisition
  - MP-4: UAV Control





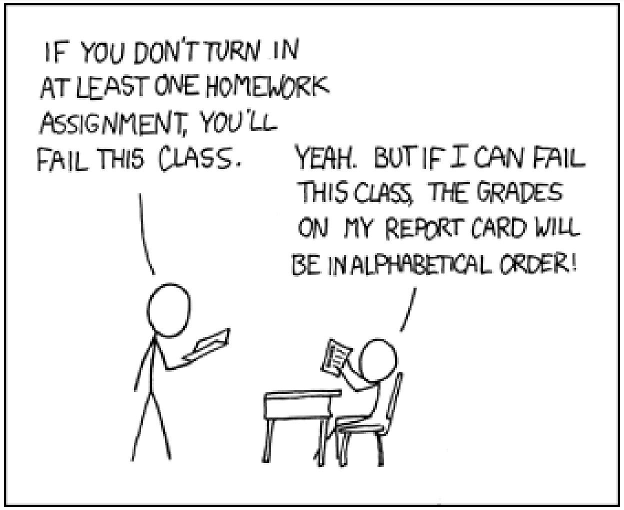
Course Project

- Student-proposed, student-assessed embedded system design project
- Essentially a capstone project – integrating your knowledge in digital logic, programming, and system design
- Something reasonable in a 5-6 week timeframe, likely leveraging existing lab infrastructure

- Deliverables:
  - Project proposal presentation and assessment rubric (week 9)
  - Project presentation and demo (10 minutes, week 16)
  - Project page on class wiki, with images / video (continuous)

Grading Policies

- Grade components:
  - Machine Problems [5x] (40%)
  - Homework (10%)
  - Class Participation (5%)
  - Midterm Exams [2x] (30%)
  - Final Project (15%)



- At first glance, CprE 488 appears to be quite a bit of work!
  - Yes. Yes it is. ☺
  - The lab/final project component is probably the most important
  - If you are a valuable member of your lab team, you will get an A

- Our goals as your instructor:
  - To create a fun, yet challenging, collaborative learning environment
  - To motivate the entire class to a 4.0 GPA
  - To inspire you to learn more (independent study / MS thesis ideas?)

Some High-Level Challenges

- How much hardware do we need?
  - How fast is the CPU? How large is Memory?
- How do we meet our deadlines?
  - Faster hardware or cleverer software?
- How do we minimize power?
  - Turn off unnecessary logic?
  - Reduce memory accesses?
  - Data compression?
- Multi-objective optimization in a vast design space

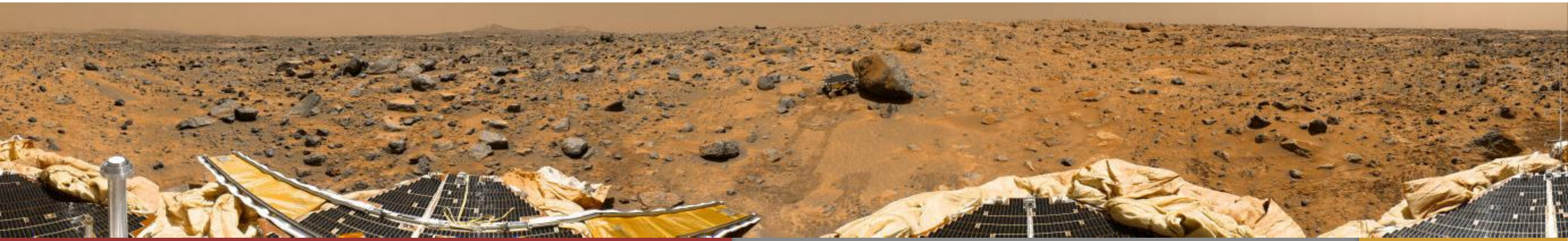
Design Considerations: Mars Rovers

Mars Sojourner Rover (1997)

- About 25 pounds
- 25 x 19 x 12 inches
- 8-bit Intel 80C85
  - 100 KHz

Opportunity/ Spirit (2004)

- About 400 pounds
- 5.2 x 7.5 x 4.9 ft
- 32-bit Rad6000
  - 20 MHz
  - cost: ??



Some High-Level Challenges

- How much hardware do we need?
  - How big is the CPU? Memory?
- How do we meet our deadlines?
  - Faster hardware or cleverer software?
- How do we minimize power?
  - Turn off unnecessary logic?
  - Reduce memory accesses?
  - Data compression?
- Multi-objective optimization in a vast design space

Some High-Level Challenges

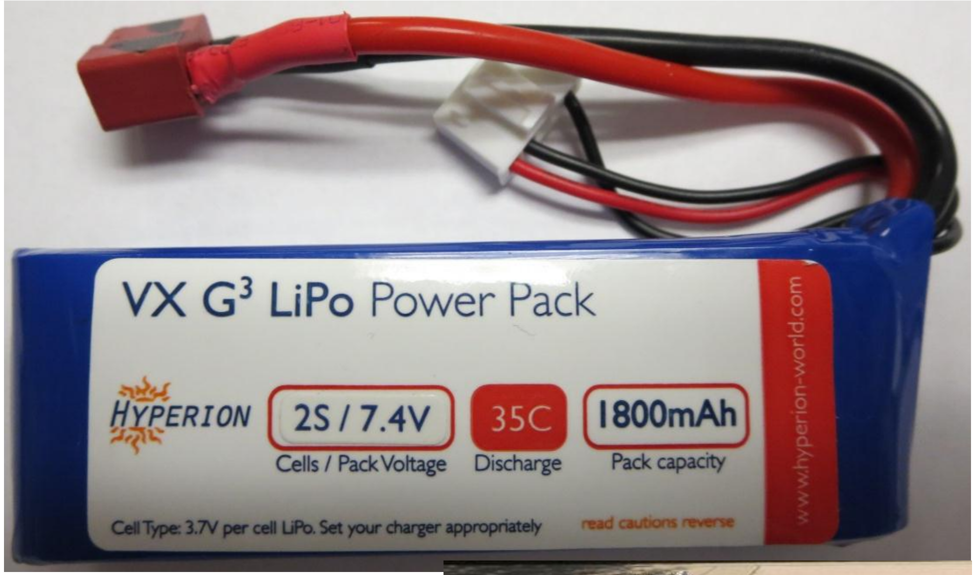
- How much hardware do we need?
  - How big is the CPU? Memory?
- How do we meet our deadlines?
  - Faster hardware or cleverer software?
- How do we minimize power?
  - Turn off unnecessary logic?
  - Reduce memory accesses?
  - Data compression?
- Multi-objective optimization in a vast design space

System Constraints and Considerations

- Exploring Martian surface (Power consumption)
  - Movement
  - Communications
  - Computation

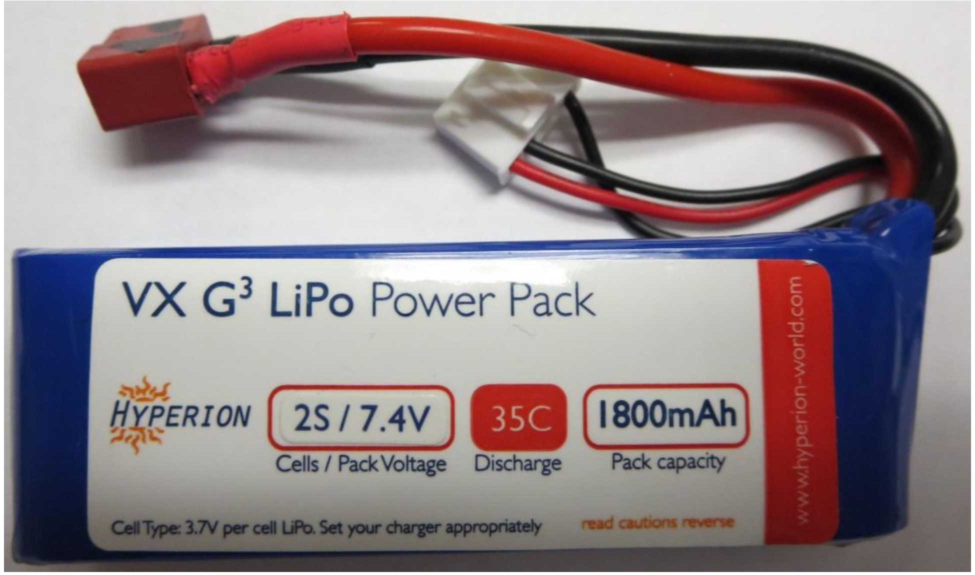
Energy / Power

- Quadcopter Battery
  - Capacity: ~2000mAh
  - Max current: 35C, 7.4V
  - Quad
    - On average requires 20A
    - Average Watts required?
    - Average Flight time?



Energy / Power

- Quadcopter Battery
  - Capacity: ~2000mAh
  - Max current: 35C, 7.4V
  - Quad
    - On average requires 20A
    - 20A * 7.4V = 148 W
    - 2000/20,000=.1 hr=6 min



System Constraints and Considerations

- Exploring Martian surface (Power consumption)
  - Movement: (??)
  - Communications: (??)
  - Computation: (??)
- Power Available
  - Solar panels (140W, 4-hours/day)
  - Battery storage

System Constraints and Considerations

- Exploring Martian surface (Power consumption)
  - Movement: 100 W
  - Communications: Rover-Orbiter (5W), Rover-Earth (100W)
  - Computation: 20W@20Mhz, 5W@2.5MHz
- Power Available
  - Solar panels (140W, 4-hours/day)
  - Battery storage

System Constraints and Considerations

- Exploring Martian surface (Power consumption)
  - Movement: 100 W
  - Communications: Rover-Orbiter (5W), Rover-Earth (100W)
  - Computation: 20W@20Mhz, 5W@2.5MHz
- Power Available
  - Solar panels (140W, 4-hours/day)
- Capabilities
  - 3,500 – 12,000 bit/s to Earth
  - ~120,000 bit/s to Orbiter

System Constraints and Considerations

- Exploring Martian surface (Power consumption)
  - Movement: 100 W
  - Communications: Rover-Orbiter (5W), Rover-Earth (100W)
  - Computation: 20W@20Mhz, 5W@2.5MHz
- Power Available
  - Solar panels (140W, 4-hours/day)
- Capabilities
  - 3,500 – 12,000 bit/s to Earth
  - ~120,000 bit/s to Orbiter
- Task
  - Image transmission:1024x1024 12-bit-pixels

Image transmission

- Communicating with Earth or Orbiter
  - Communications: Rover-Orbiter (5W), Rover-Earth (100W)
    - 100,000 bits/s to Obiter 10,000 bit/s to Earth
  - Computation: 20W@20Mhz, 5W@2.5MHz
  - Image Size: 1000x1000 10-bit-pixels
- Constraints
  - 3 hour window/day for Earth transmission
  - 10 min widow/day for Obiter transmission
- Compute
  - Time to send 1 image to Earth, to Obiter
  - How many pics per day (Earth and Obiter)

Image transmission

- Communicating with Earth or Orbiter
  - Communications: Rover-Orbiter (5W), Rover-Earth (100W)
    - 100,000 bits/s to Obiter 10,000 bit/s to Earth
  - Computation: 20W@20Mhz, 5W@2.5MHz
  - Image Size: 1000x1000 10-bit-pixels
- Constraints
  - 3 hour window/day for Earth transmission
  - 10 min widow/day for Obiter transmission
- Compute
  - Time to send 1 image to Earth, to Obiter
  - How many pics per day (Earth and Obiter)

Channel	Power (W=J/s)	Time/ pic (s)	Energy/ pic (J)	Time/ day (s)	Pics /day	Energy /day (J)		
Rov->Orb	5			600				
Rov->Earth	100			10,000				

Image transmission

- Communicating with Earth or Orbiter
  - Communications: Rover-Orbiter (5W), Rover-Earth (100W)
    - 100,000 bits/s to Obiter 10,000 bit/s to Earth
  - Computation: 20W@20Mhz, 5W@2.5MHz
  - Image Size: 1000x1000 10-bit-pixels
- Constraints
  - 3 hour window/day for Earth transmission
  - 10 min widow/day for Obiter transmission
- Compute
  - Time to send 1 image to Earth, to Obiter
  - How many pics per day (Earth and Obiter)

Channel	Power (W=J/s)	Time/ pic (s)	Energy/ pic (J)	Time/ day (s)	Pics /day	Energy /day (J)		
Rov->Orb	5	100	500	600	6	3,000		
Rov->Earth	100	1,000	100,000	10,000	10	1,000,000		

Image transmission

- Communicating with Earth or Orbiter **(5,000 J / day budget)**
  - Communications: Rover-Orbiter (5W), Rover-Earth (100W)
    - 100,000 bits/s to Obiter 10,000 bit/s to Earth
  - Computation: 20W@20Mhz, 5W@2.5MHz
  - Image Size: 1000x1000 10-bit-pixels
- Constraints
  - 3 hour window/day for Earth transmission
  - 10 min widow/day for Obiter transmission
- Compute
  - Time to send 1 image to Earth, to Obiter
  - How many pics per day (Earth and Obiter)

Channel	Power (W=J/s)	Time/ pic (s)	Energy/ pic (J)	Time/ day (s)	Pics /day	Energy /day (J)		
Rov->Orb	5	100	500	600	6	3,000		
Rov->Earth	100	1,000	100,000	10,000	10	1,000,000		

Image transmission

- Communicating with Earth or Orbiter **(5,000 J / day budget)**
  - Communications: Rover-Orbiter (5W), Rover-Earth (100W)
    - 100,000 bits/s to Obiter 10,000 bit/s to Earth
  - Computation: 20W@20Mhz, 5W@2.5MHz
  - Image Size: 1000x1000 10-bit-pixels
- Constraints
  - 3 hour window/day for Earth transmission
  - 10 min widow/day for Obiter transmission
- How could you get a better image rate?

Channel	Power (W=J/s)	Time/ pic (s)	Energy/ pic (J)	Time/ day (s)	Pics /day	Energy /day (J)		
Rov->Orb	5	100	500	600	6	3,000		
Rov->Earth	100	1,000	100,000	10,000	10	1,000,000		

Image transmission

- Communicating with Earth or Orbiter **(5,000 J / day budget)**
  - Communications: Rover-Orbiter (5W), Rover-Earth (100W)
    - 100,000 bits/s to Obiter (10 min), 10,000 bit/s to Earth (3 hr)
  - Computation: 20W@20Mhz, 5W@2.5MHz
  - Image Size: 1000x1000 10-bit-pixels=10,000,000 bits/image
- Compression: ICER (Compression Incremental cost-effectiveness Ratio): ~1 bit/pixel
  - 1,000,000 bits/image
- Compress 1/pixel per clock.
  - How long to compress 1 image?
  - How much Energy to compress 1 image?

Channel	Power (W=J/s)	Time/ pic (s)	Energy/ pic (J)	Time/ day (s)	Pics /day	Energy /day (J)	Comp time /pic (s)	Comp Eng /pic (J)
Rov->Orb	5	100	500	600	6	3,000		
Rov->Earth	100	1,000	100,000	10,000	10	1,000,000		

Image transmission

- Communicating with Earth or Orbiter **(5,000 J / day budget)**
  - Communications: Rover-Orbiter (5W), Rover-Earth (100W)
    - 100,000 bits/s to Obiter (10 min), 10,000 bit/s to Earth (3 hr)
  - Computation: 20W@20Mhz, 5W@2.5MHz
  - Image Size: 1000x1000 10-bit-pixels=10,000,000 bits/image
- Compression: ICER (Compression Incremental cost-effectiveness Ratio): ~1 bit/pixel
  - 1,000,000 bits/image
- Compress 1/pixel per clock.
  - How long to compress 1 image?
  - How much Energy to compress 1 image?

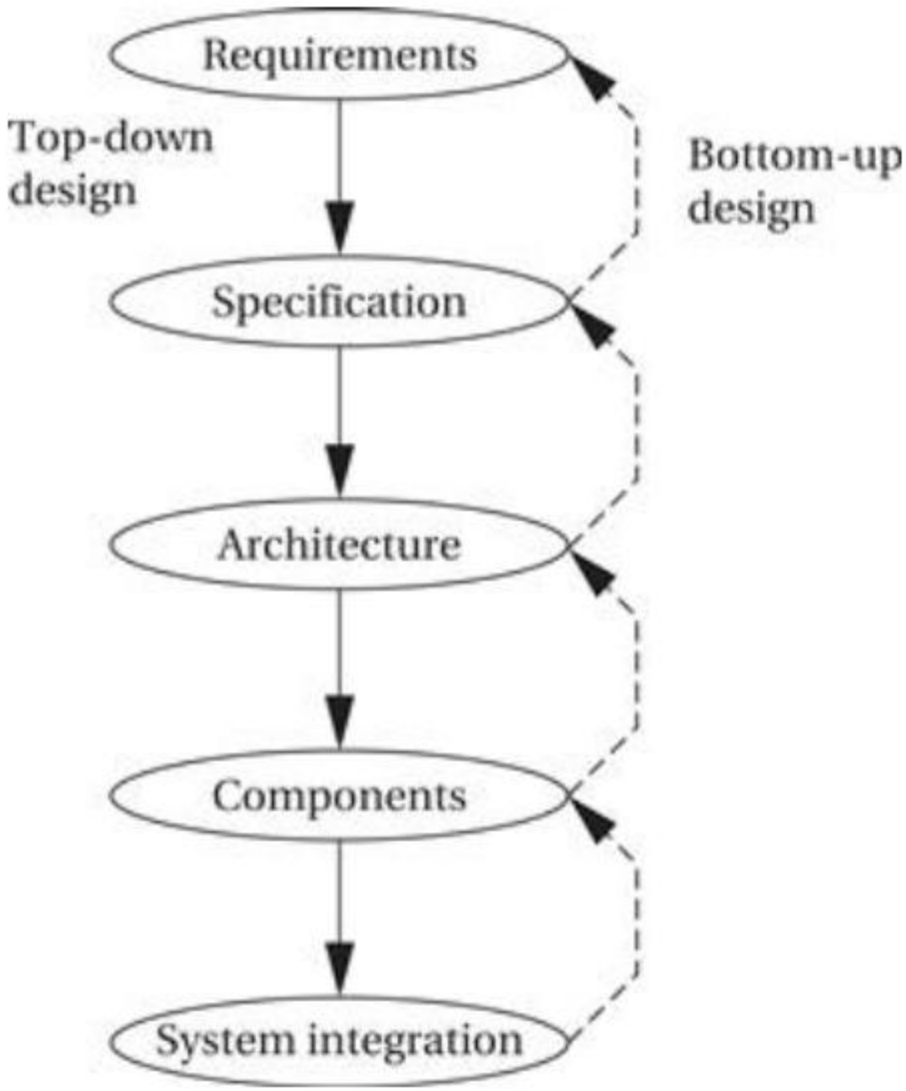
Channel	Power (W=J/s)	Time/ pic (s)	Energy/ pic (J)	Time/ day (s)	Pics /day	Energy /day (J)	Comp time /pic (s)	Comp Eng /pic (J)
Rov->Orb	5	100	500	600	6	3,000	.05 / .4	1 / 2
Rov->Earth	100	1,000	100,000	10,000	10	1,000,000	.05 / .4	1 / 2

Illustrative Design Exercise

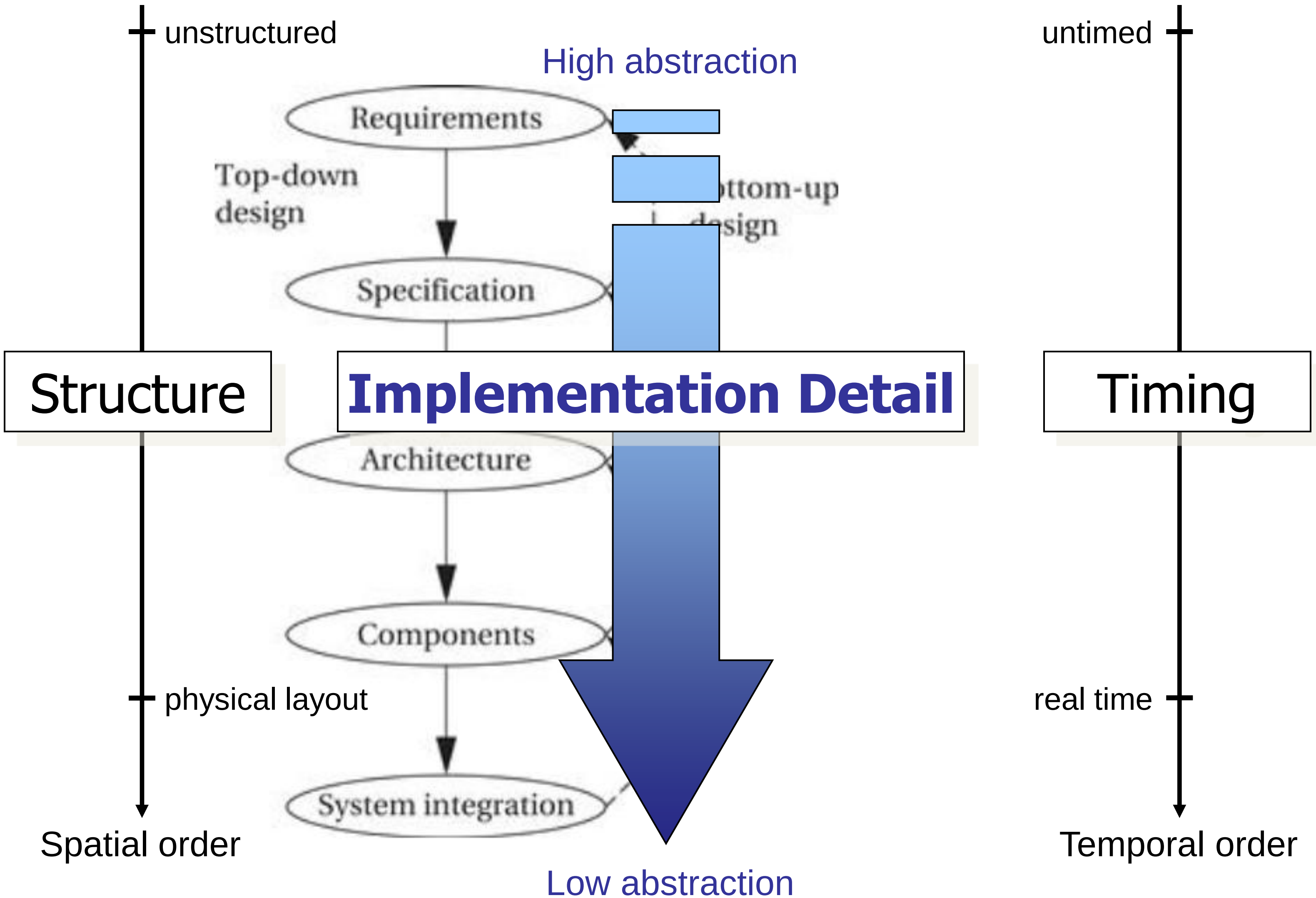
- An illustrative example of embedded system design inspired by Chapter 1 of the M. Wolf textbook

GPS Navigation Unit

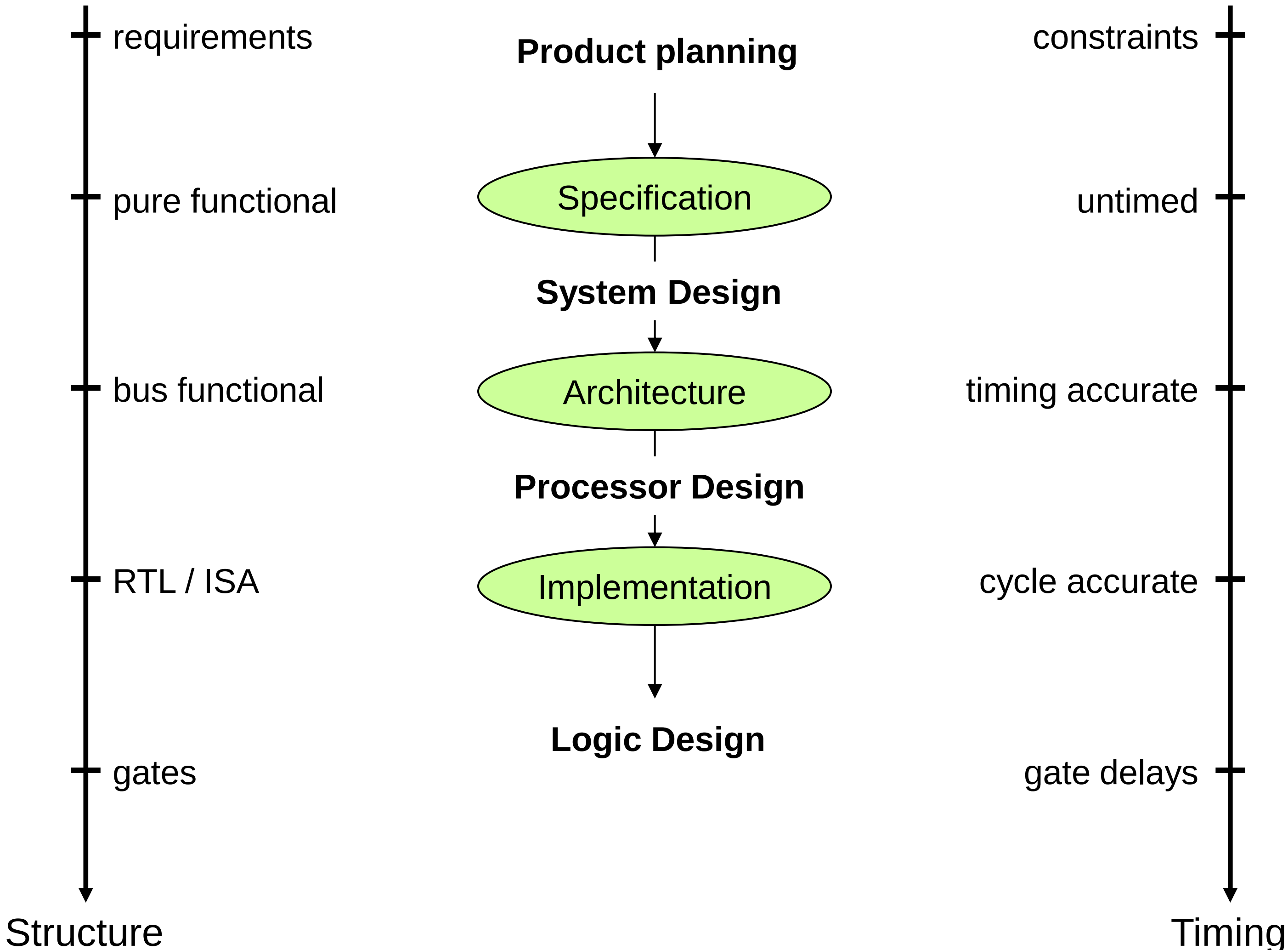
Major Steps in the Design Process



Abstraction Levels

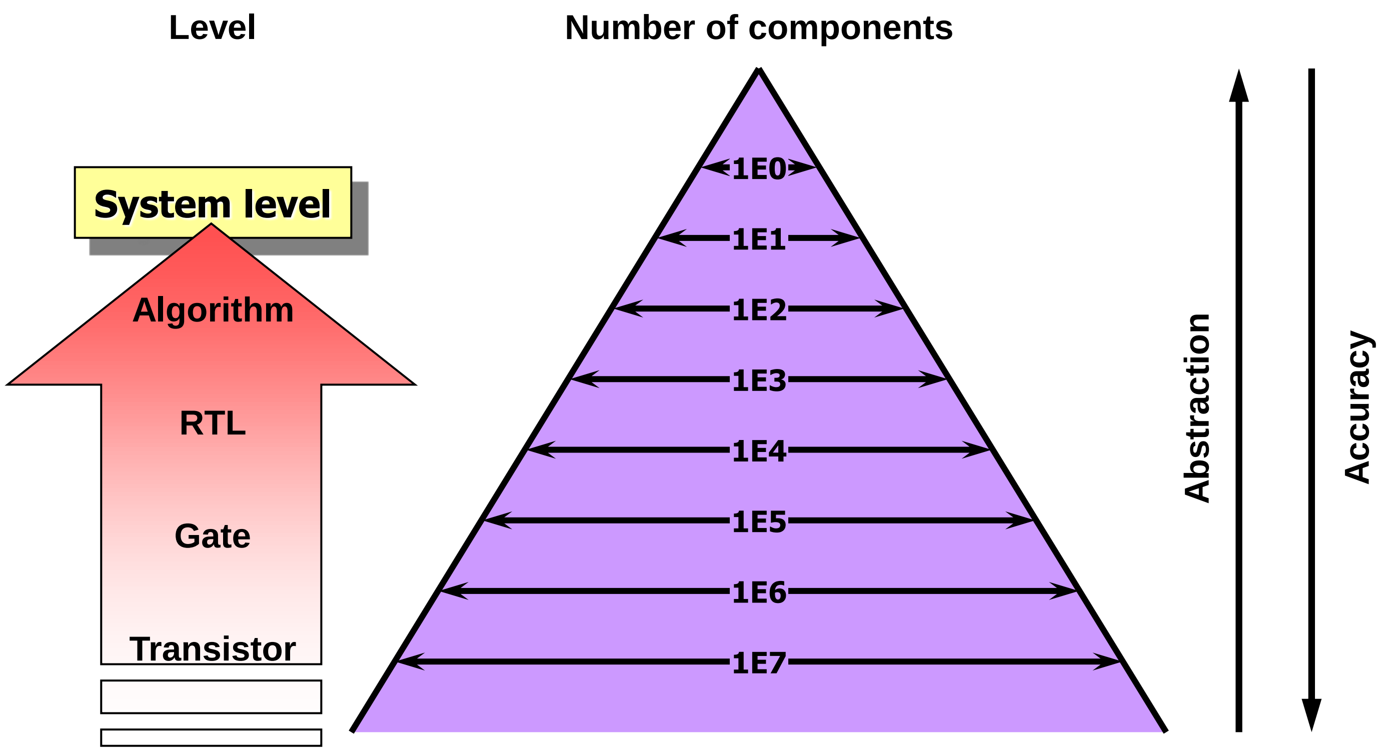


Abstraction Levels [cont]



Abstraction Levels [cont]

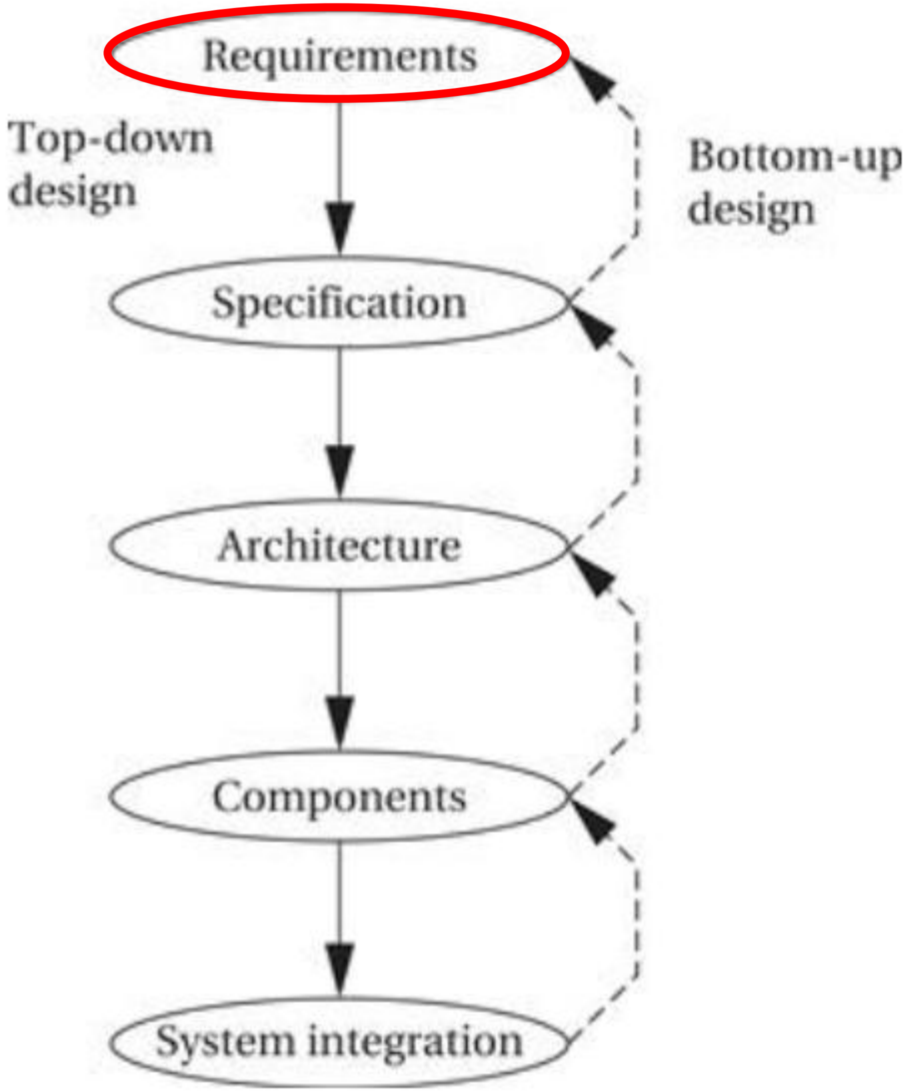
- Growing system complexities
- ➢ Move to higher levels of abstraction [ITRS07, itrs.net]
  - ➢ Electronic system-level (ESL) design



Source: R. Doemer, UC Irvine

Major Steps in the Design Process

GPS Navigation Unit



Requirements

- Plain language description of what the user wants and expects to get
- May be developed in several ways:
  - Talking directly to customers (User Research)
  - Talking to marketing representatives
  - Providing prototypes to users for comment

Functional vs. Non-Functional Requirements

- Functional requirements:
  – output as a function of input
- Non-functional requirements:
  – time required to compute output;
  – size, weight, etc.;
  – power consumption;
  – reliability;
  – etc.

GPS Navigation Unit Requirements

- Example: Table for summarizing metrics of interest

Name	GPS moving map
Purpose	
Inputs	
Outputs	
Functions	
Performance	
Manufacturing cost	
Power	
Physical size and weight	

GPS Navigation Unit Requirements

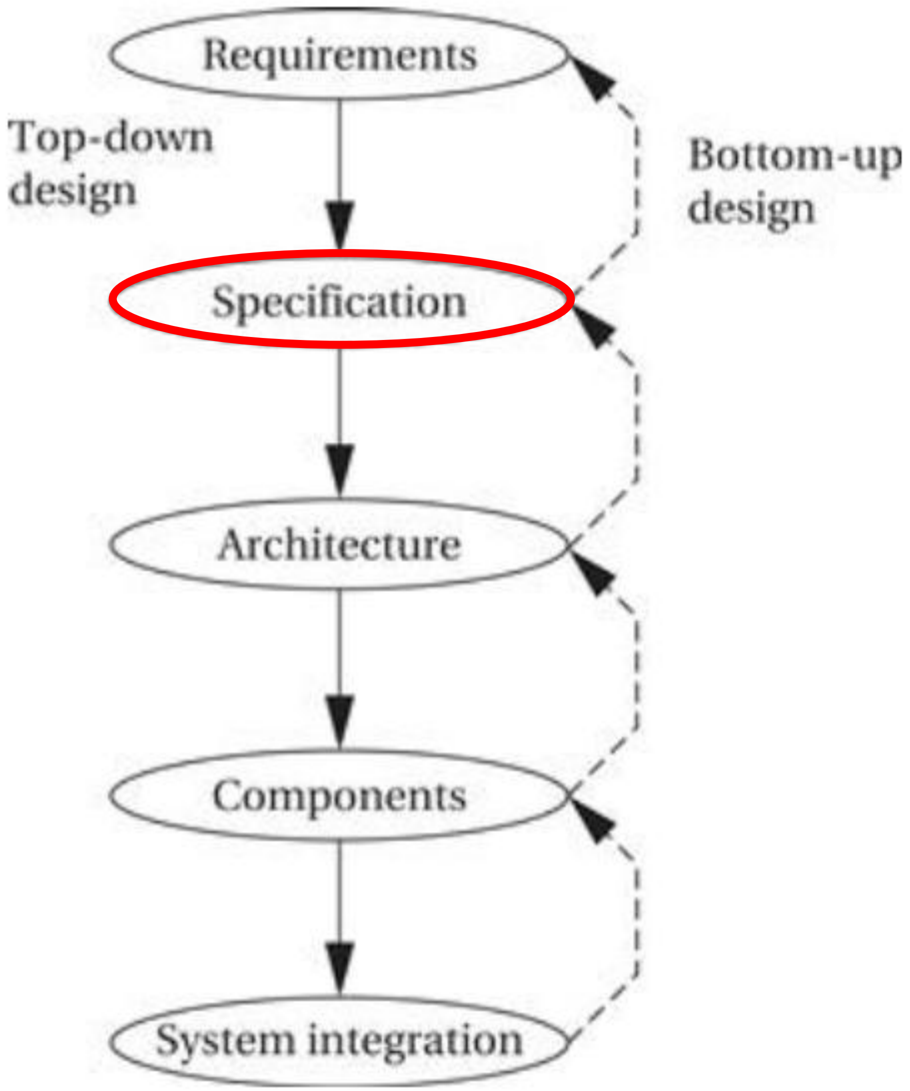
- Functionality: Hand held. Show major roads & landmarks.
- User interface: At least 400 x 600 pixel screen. Three buttons max. Pop-up menu.
- Performance: Map should scroll smoothly. No more than 1 sec power-up. Lock onto GPS within 15 seconds.
- Cost: $120 street price = approx. $40 cost of goods sold.
- Physical size/weight: Should fit in hand
- Power consumption: Should run for 8 hours on four AA batteries
- Any others?

Requirements: Summary & Prototype

Name	GPS moving map
Purpose	Consumer-grade moving map for driving use
Inputs	Power button, two control buttons
Outputs	Back-lit LCD display 400 × 600
Functions	Uses 5-receiver GPS system; three user-selectable resolutions; always displays current latitude and longitude
Performance	Updates screen within 0.25 seconds upon movement
Manufacturing cost	$40
Power	100 mW
Physical size and weight	No more than 2″ × 6″, 12 ounces

Major Steps in the Design Process

GPS Navigation Unit

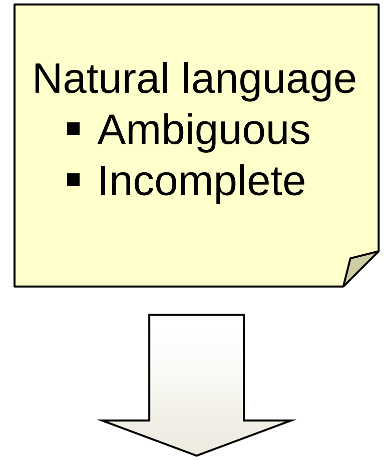


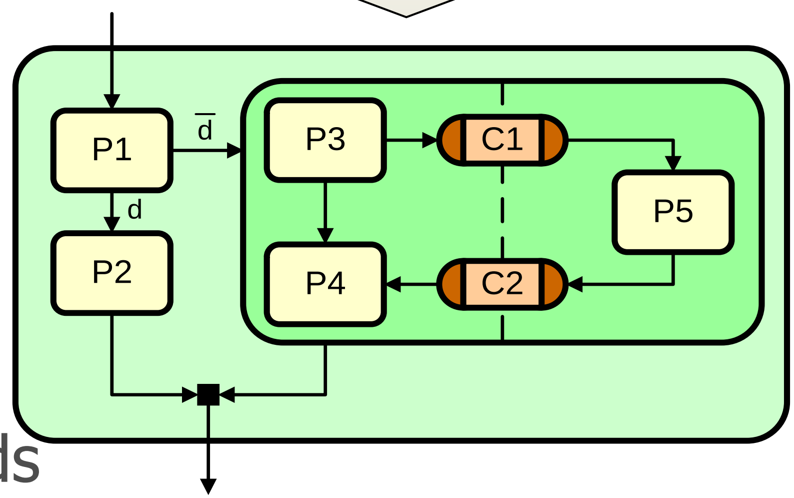
GPS Specification

- What should system include:
  - What is received from GPS;
  - Map data;
  - User interface;
  - Operations required to satisfy user requests;
  - Background operations needed to keep the system running
- Often described using mechanisms such as:
  - UML
  - Data/Control Flow diagrams, Compute Model (FSM)
  - Formal Method language (1^{st} order logic, LTL)

System Specification

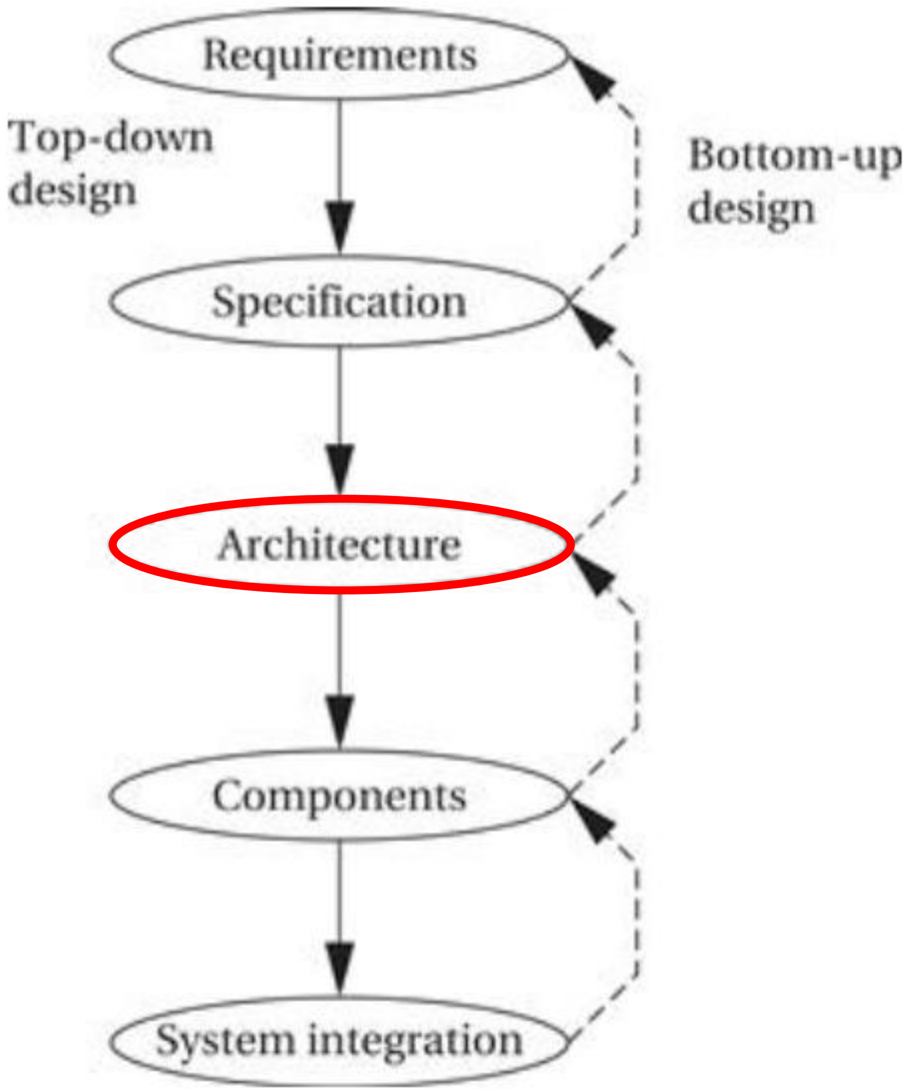
- Capture requirements
  - Functional
    - Free of any implementation details
  - Non-functional
    - Quality metrics, constraints
- Formal representation
  - Models of computation
    - Allow analysis of properties
  - Executable
    - Can validate using simulation
    - Can verify with formal methods
- Used for application development
  - Precise description of desired system behavior





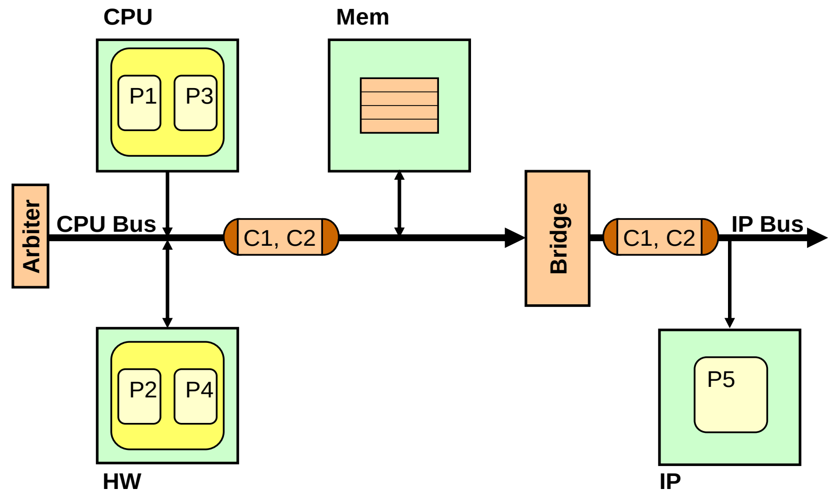
Major Steps in the Design Process

GPS Navigation Unit



System Architecture

- Processing elements (PEs)
  - Processors
    - General-purpose, programmable
    - Digital signal processors (DSPs)
    - Application-specific instruction set processor (ASIP)
    - Custom hardware processors
    - Intellectual property (IP)
  - Memories

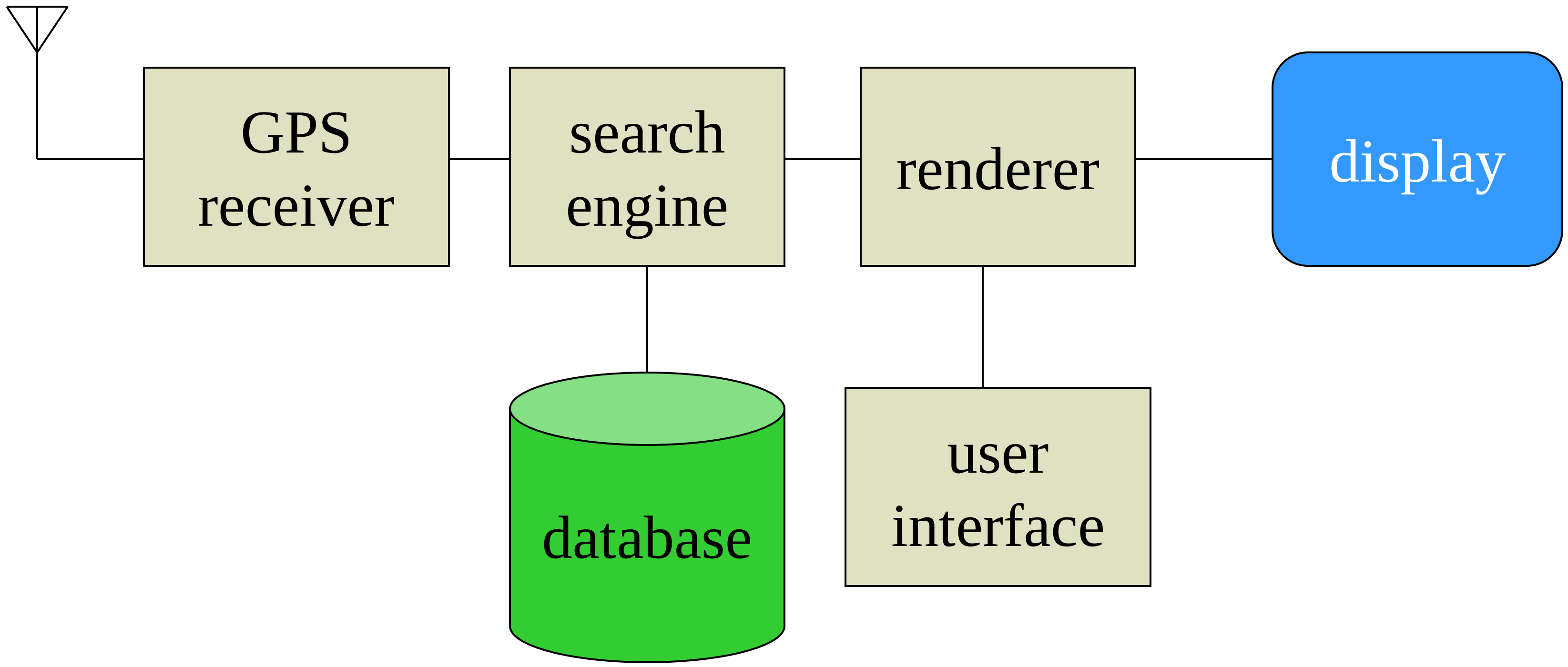


- Communication elements (CEs)
  - Transducers, bus bridges
  - I/O peripherals
- Busses
  - Communication media
    - Parallel, master/slave protocols
    - Serial and network media

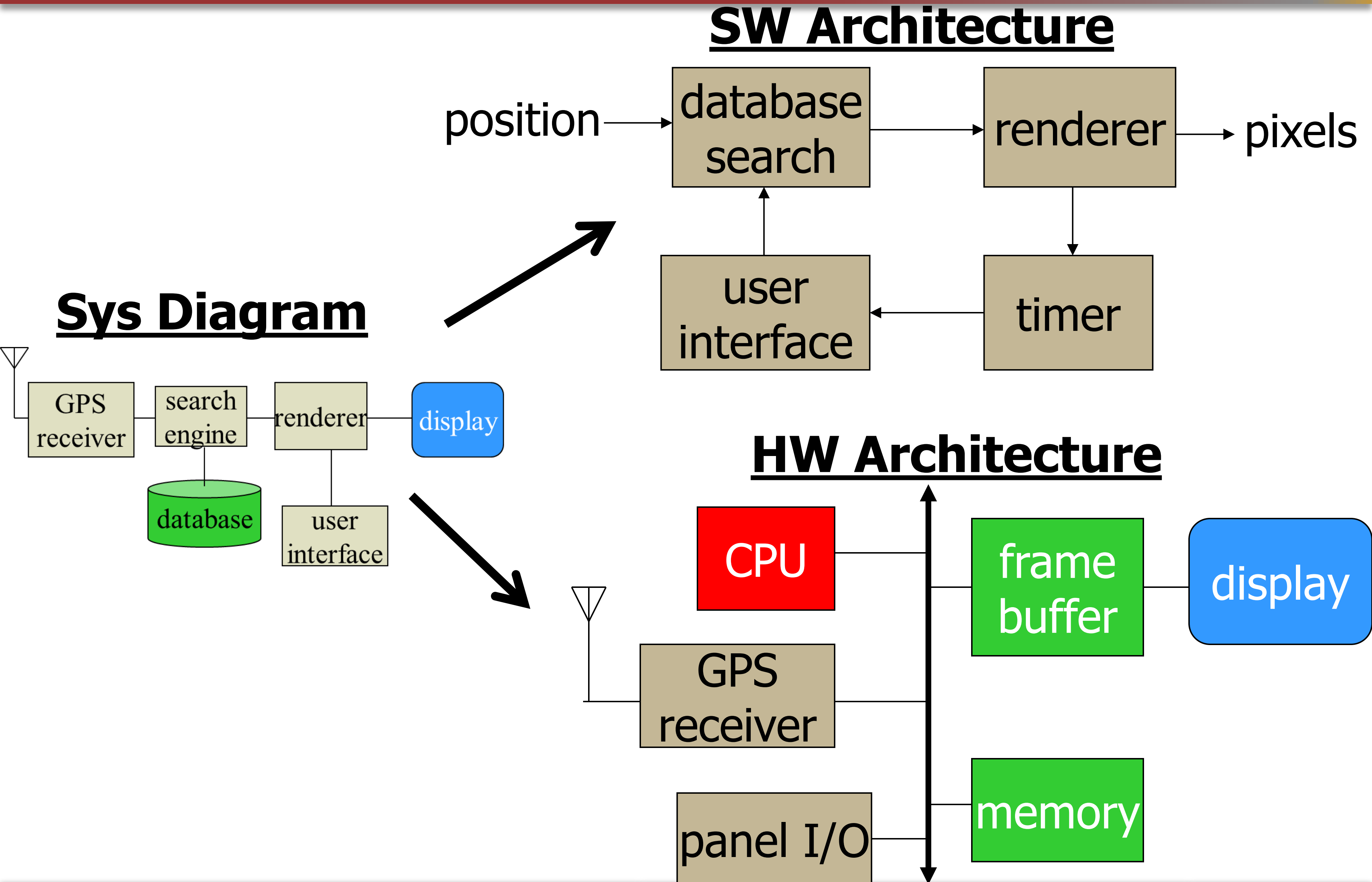
➢ Heterogeneous multi-processor systems

  ➢ Multi-Processor System-on-Chip (MPSoC)

GPS Unit System Architecture (Diagram)

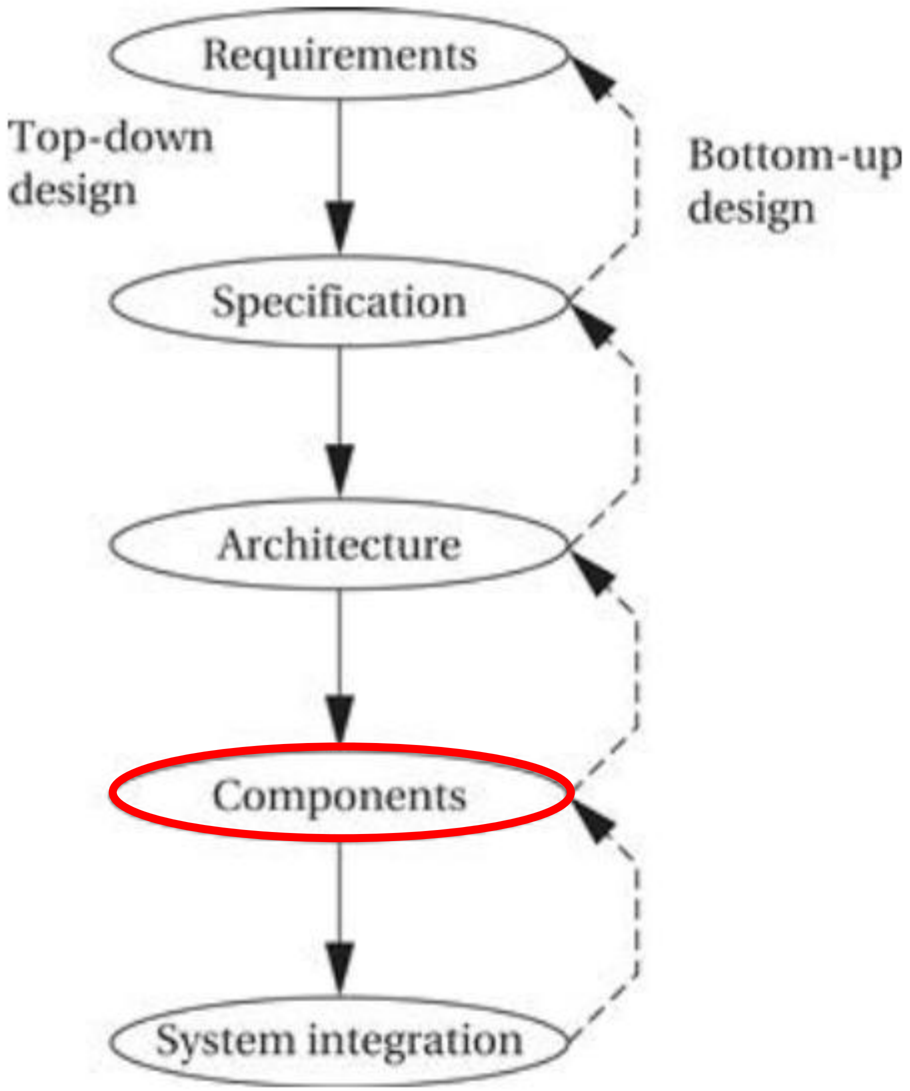


GPS Unit Architecture



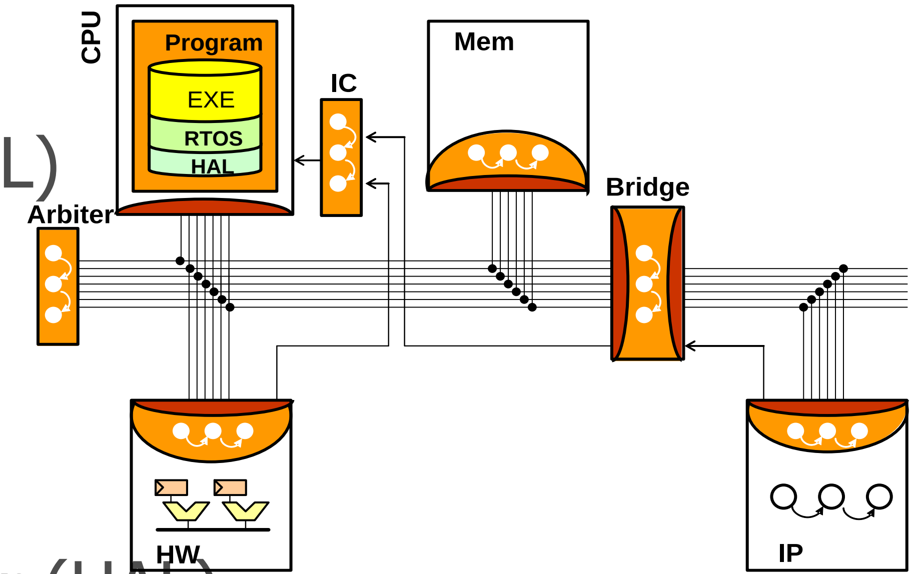
Major Steps in the Design Process

GPS Navigation Unit



Component Design/Implementation

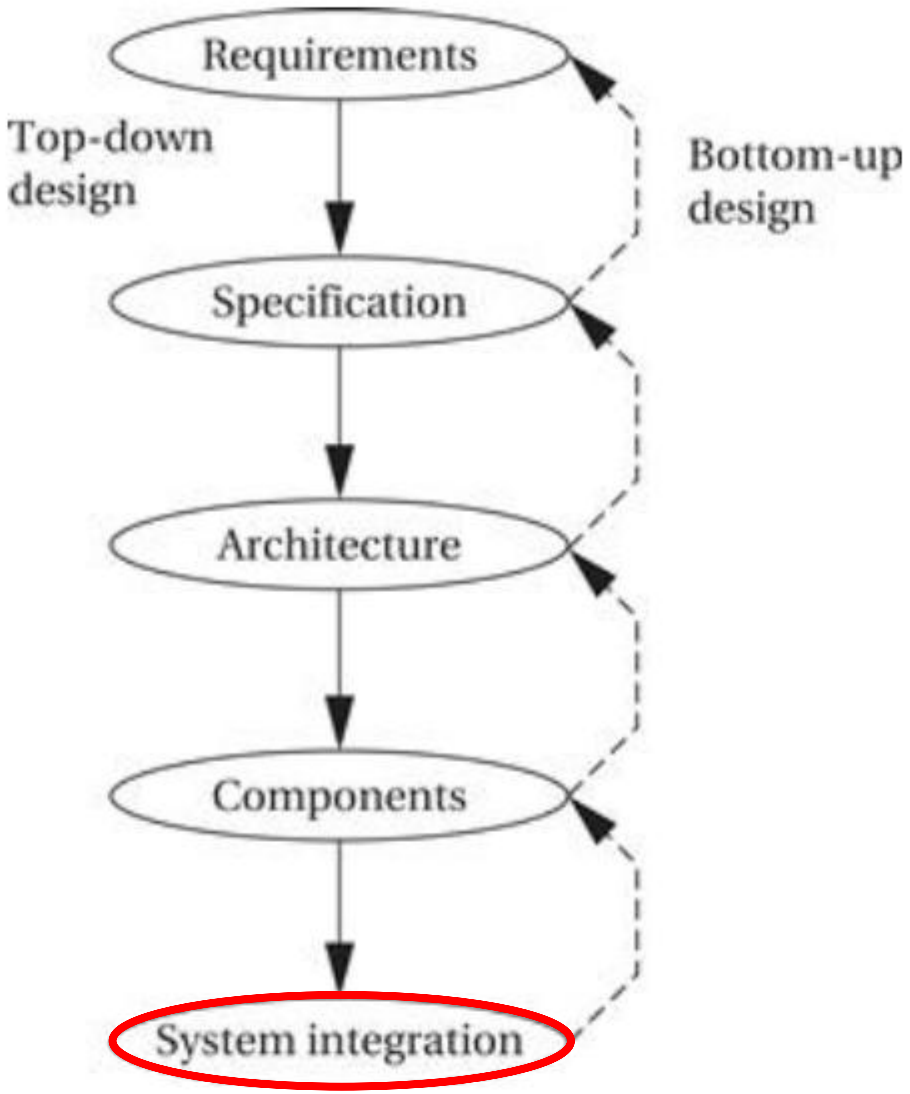
- Hardware
  - Microarchitecture
  - Register-transfer level (RTL)
- Software binaries
  - Application object code
  - Real-time operating system (RTOS)
  - Hardware abstraction layer (HAL)
- Interfaces
  - Pins and wires
  - Arbiters, muxes, interrupt controllers (ICs), etc.
  - Bus protocol state machines



- ➢ Further logic and physical synthesis
  - ➢ Manufacturing
  - ➢ Prototyping boards

Major Steps in the Design Process

GPS Navigation Unit



Component Design and System Integration

- Must spend time architecting the system before coding
  - Draw pictures/diagrams at various levels of detail
- Evaluate Component Sourcing Options:
  - Ready-made,
  - Modified from existing designs,
  - Designed from scratch
- Putting components together early
  - Many bugs appear only at this stage
- Have a plan for integrating components to uncover bugs quickly, test as much functionality as early as possible

Important questions to keep in mind

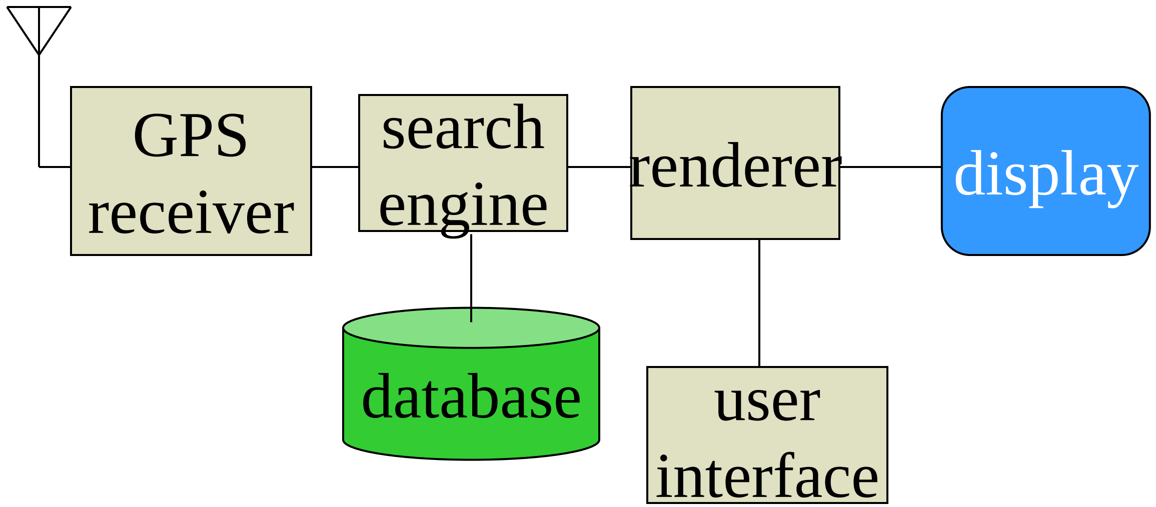
- Does it really work?
  - Is the specification correct?
  - Does the implementation meet the spec?
  - How do we test for real-time characteristics?
  - How do we test on real data?
- How do we work on the system?
  - Observability, controllability?
  - What is our development platform?

Acknowledgments

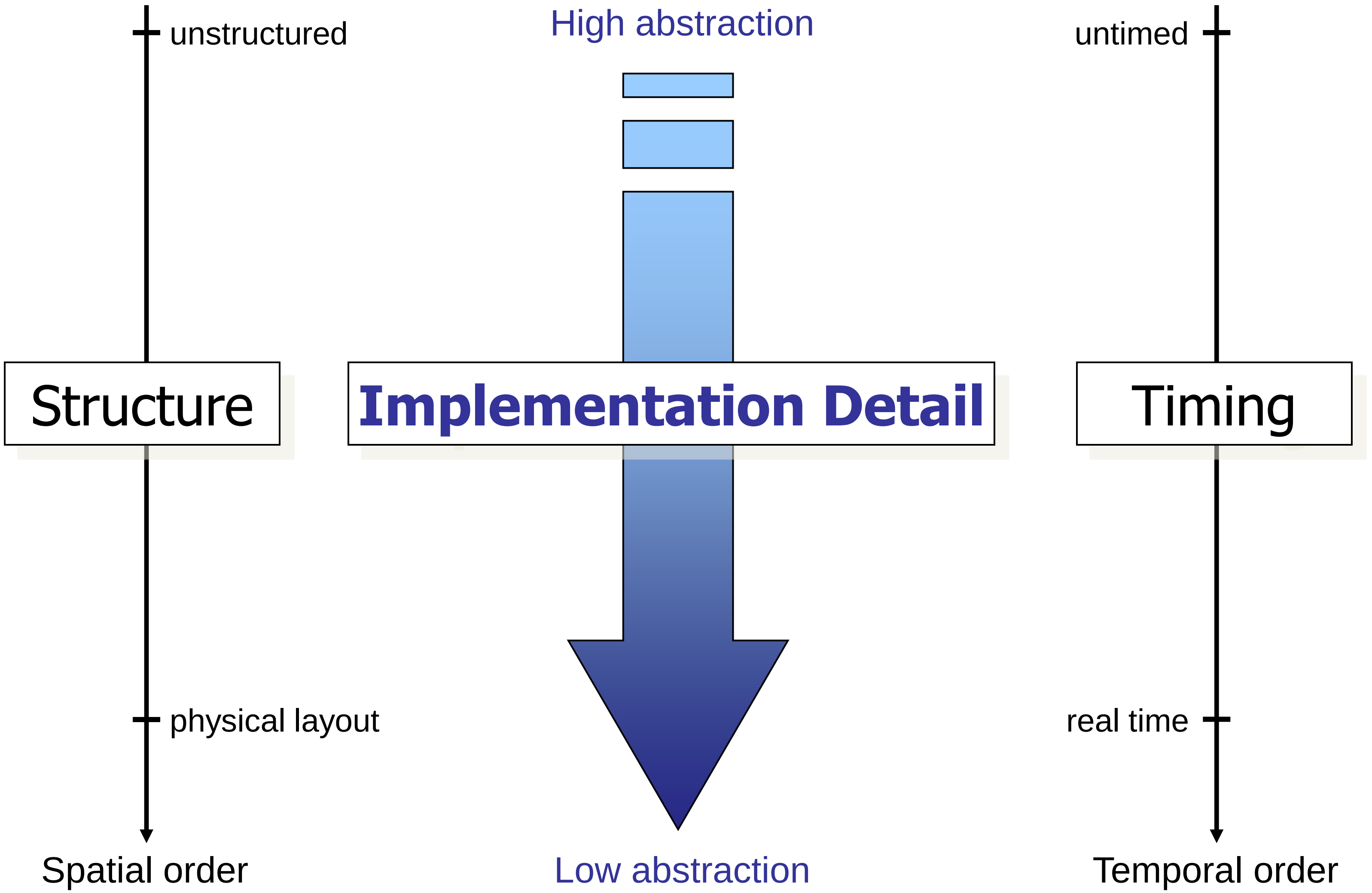
- These slides are inspired in part by material developed and copyright by:
  - Marilyn Wolf (Georgia Tech)
  - Frank Vahid (UC-Riverside)
  - A. Gerstlauer (UT-Austin)
  - Daniel Gajski (UC-Irvine)
  - Ed Lee (UC-Berkeley)
  - James Hamblen (Georgia Tech)

Extra slides

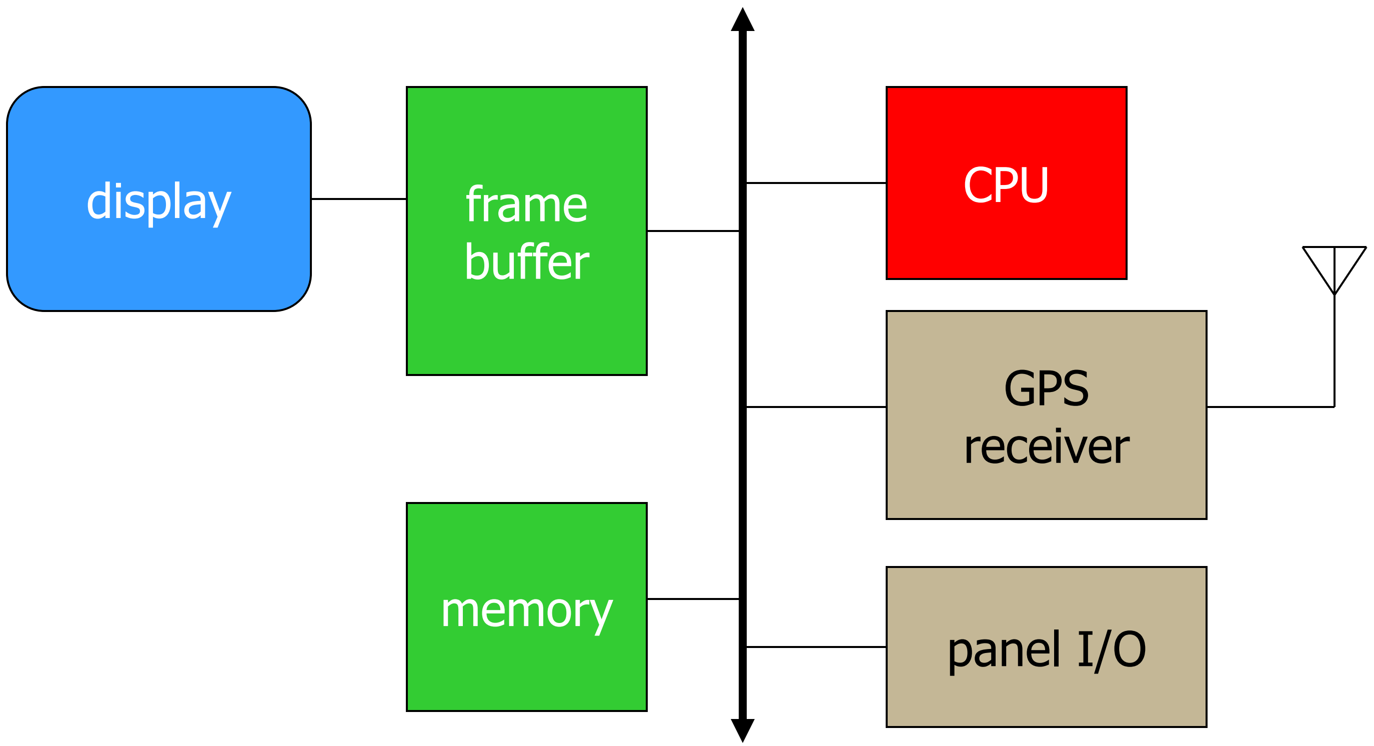
GPS Unit System Specification (Diagram)



Abstraction Levels [cont.]



GPS Unit System Architecture (HW)



GPS Unit System Specification (SW)