嵌入式微處理機系統

01 System Introduction

國立台灣科技大學電機系
王乃堅老師
Feb. 2024

Introduction

- Topics:
  - What are embedded systems?
  - Challenges in embedded computing system design.
  - Design methodologies.

Introduction

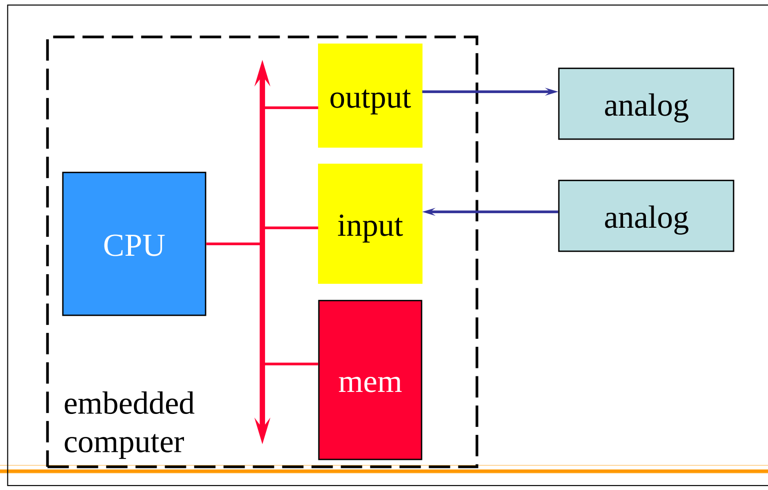
- Embedded system:
  - any device that includes a programmable computer but is not itself a general-purpose computer.
  - Take advantage of application characteristics to optimize the design:
  - don’t need all the general-purpose bells and whistles.

Introduction

- What is the “Embedded System”?
  - 以應用為中心，以計算機為基礎
  - 軟硬體可修改，適用於應用系統對功能、可靠性、成本、體積、功耗有嚴格要求的專用計算機系統。
  - 依據英國電機工程師協會的定義
    - 嵌入式系統為控制、監視或輔助設備、機器或甚至工廠運作的裝置。
  - 嵌入式系統一般指非PC系統
    - 硬體包括處理器／微處理器、記憶體及外設器件和I／O埠、圖形控制器等。
    - 軟體部分包括作業系統軟體（OS）（要求即時和多工操作）和應用程式編程。

Embedding a computer

- What is the “Embedded System”?



Early History (1/2)

- Late 1940’s: MIT Whirlwind computer was designed for real-time operations.
  - Originally designed to control an aircraft simulator.
- First microprocessor was Intel 4004 in early 1970’s.
- HP-35 calculator used several chips to implement a microprocessor in 1972.

Early History (2/2)

- Automobiles used microprocessor-based engine controllers starting in 1970's.
  - Control fuel/air mixture, engine timing, etc.
  - Multiple modes of operation:
    - warm-up,
    - cruise,
    - hill climbing, etc.
  - Provides lower emissions, better fuel efficiency.

Microprocessor varieties

- Microcontroller:
  - includes I/O devices, on-board memory.
- Digital signal processor (DSP):
  - microprocessor optimized for digital signal processing.
- Typical embedded word sizes: 8-bit, 16-bit, 32-bit.

Applications

- Personal digital assistant (PDA)
- Cell phone, iphone , …
- Automobile: engine, brakes, dash, etc.
- Television, Set-Top-Box
- GPS
- Printer
- PC keyboard (scans keys)
- Household appliances, MP3 player, …

Application Examples

- Simple control: front panel of microwave oven, etc.
- Canon EOS 3 has three microprocessors.
  - 32-bit RISC CPU runs autofocus and eye control systems.
- Analog TV: channel selection, etc.
- Digital TV: programmable CPUs + hardwired logic.

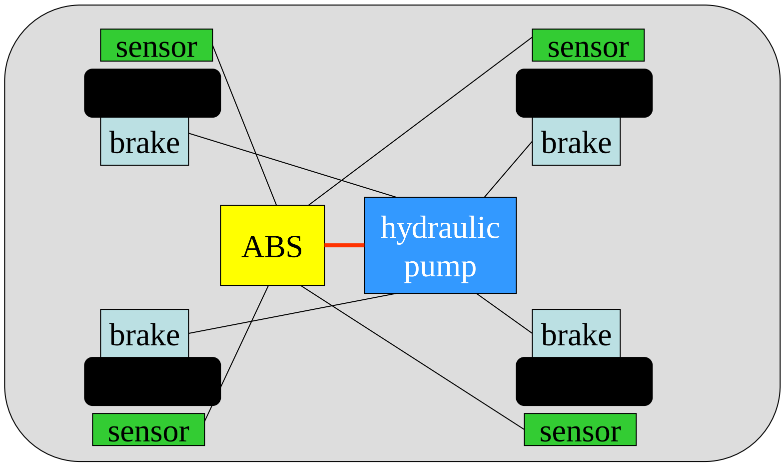
Automotive embedded systems

- Today's high-end automobile may have 100 microprocessors:
  - 4-bit microcontroller checks seat belt;
  - microcontrollers run dashboard devices;
  - 16/32-bit microprocessor controls engine.

BMW 850i brake and stability control system

- Anti-lock brake system (ABS)
  - pumps brakes to reduce skidding.
- Automatic stability control (ASC+T)
  - controls engine to improve stability.
- ABS and ASC+T communicate
  - ABS was introduced first --- needed to interface to existing ABS module.

BMW 850i, cont'd.



Characteristics of embedded systems

- Sophisticated functionality.
- Real-time operation.
- Low manufacturing cost.
- Low power.
- Designed to tight deadlines by small teams.

嵌入式系統特點 (1/2)

- 特定用戶群：
  - 低功耗、體積小、集成度高等特點，小型化，移動能力增強，跟網路的耦合也越來越緊密。
- 功耗很低：
  - 可攜式的無線及移動的計算和通信設備中靠電池供電，需要功耗只有mW~μW級
- 不斷創新的知識集成系統：
  - 結合電腦、半導體和電子技術與各個行業的具體應用相後的產物。是技術密集、資金密集、高度分散。

嵌入式系統特點 (2/2)

- 高效率地設計：
  - 硬體和軟體都必須量體裁衣、去除冗餘，力爭在同樣的矽片面積上實現更高的性能
- 生命週期較長：
  - 和具體應用結合在一起，它的升級換代也是和具體產品同步進行
- 為了提高執行速度和系統可靠性，其軟體一般都固化在記憶體晶片或單片機本身中，而不是存貯於磁片等載體中
- 嵌入式系統本身不具備自舉開發能力，即使設計完成以後用戶通常也是不能對其中的程式功能進行修改的，必須有一套開發工具和環境才能進行開發

Functional complexity

- Often have to run sophisticated algorithms or multiple algorithms.
  - Cell phone, laser printer.
- Often provide sophisticated user interfaces.

Real-time operation

- Must finish operations by deadlines.
  - Hard real time: missing deadline causes failure.
  - Soft real time: missing deadline results in degraded performance.
- Many systems are multi-rate:
  - must handle operations at widely varying rates.

Non-functional requirements

- Many embedded systems are mass-market items that must have low manufacturing costs.
  - Limited memory, microprocessor power, etc.
- Power consumption is critical in battery-powered devices.
  - Excessive power consumption increases system cost even in wall-powered devices.

Design teams

- Often designed by a small team of designers.
- Often must meet tight deadlines.
  - 6 month market window is common.
  - Can't miss back-to-school window for calculator, PC.

Why use microprocessors?

- Alternatives:
  - field-programmable gate arrays (FPGAs), custom logic, etc.
- Microprocessors are often very efficient:
  - can use same logic to perform many different functions.
- Microprocessors simplify the design of families of products.

The performance paradox

- Microprocessors use much more logic to implement a function than does custom logic.
- But microprocessors are often at least as fast:
  - heavily pipelined;
  - large design teams;
  - aggressive VLSI technology.

Power

- Custom logic is a clear winner for low power devices.
- Modern microprocessors offer features to help control power consumption.
- Software design techniques can help reduce power consumption.

Challenges in embedded system design (1/2)

- 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?

Challenges in embedded system design (2/2)

- 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?

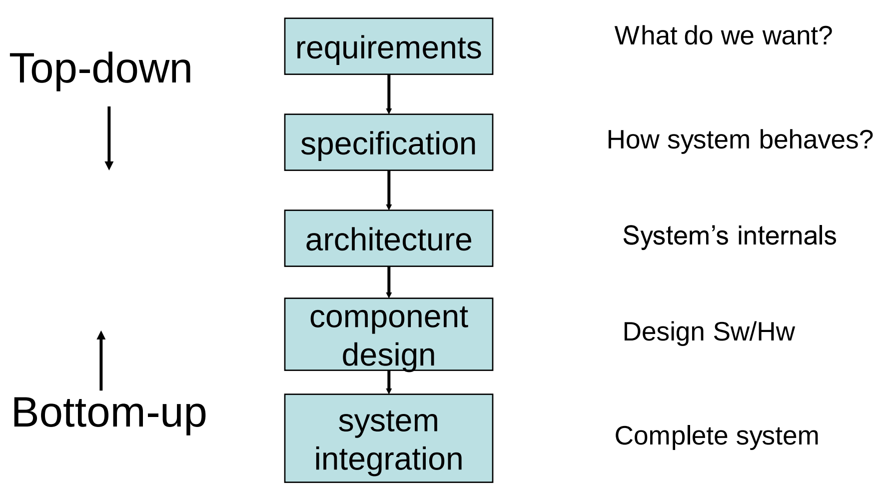
Design methodologies

- A procedure for designing a system.
- Understanding your methodology helps you ensure you didn't skip anything.
- Compilers, software engineering tools, computer-aided design (CAD) tools, etc., can be used to:
  - help automate methodology steps;
  - keep track of the methodology itself.

Design goals

- Performance.
  - Overall speed, deadlines.
- Functionality and user interface.
- Manufacturing cost.
- Power consumption.
- Other requirements (physical size, etc.)

Levels of abstraction



Top-down vs. bottom-up

- Top-down design:
  - start from most abstract description;
  - work to most detailed.
- Bottom-up design:
  - work from small components to big system.
  - When we do not have perfect insight into how later stages of the design process will turn out.
- Real design uses both techniques.

Decision at one stage of design

- Based upon estimates of what will happen later:
  - How fast can we make a particular function run?
  - How much memory will we need?
  - How much system bus capacity do we need?
- If our estimates are inadequate, we may have to backtrack and amend our original decisions to take the new facts into account.
- In general, the less experience we have, the more we will have to rely on bottom-up design.

Stepwise refinement

- At each level of abstraction, we must:
  - analyze the design to determine characteristics of the current state of the design;
  - refine the design to add detail.

Requirements

- Plain language description of what the user wants and expects to get.
- May be developed in several ways:
  - talking directly to customers;
  - talking to marketing representatives;
  - providing prototypes to users for comment.

Functional vs. non-functional requirements

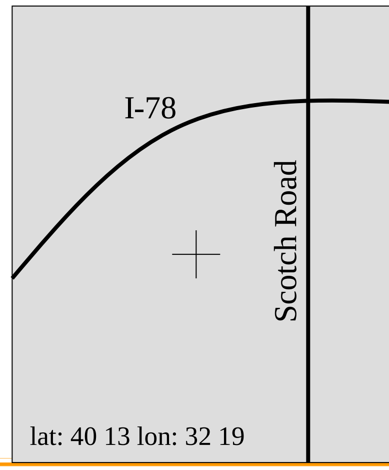
- Functional requirements:
  - output as a function of input.
- Non-functional requirements:
  - time required to compute output;
  - size, weight, etc.;
  - power consumption;
  - reliability;
  - etc.

A sample requirements form

name
purpose
inputs
outputs
functions
performance
manufacturing cost
power
physical size/weight

Example: GPS moving map requirements

- Moving map obtains position from GPS, paints map from local database.



GPS moving map needs

- Functionality: For automotive use. Show major roads and 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: $500 street price
- Physical size/weight: Should fit in hand.
- Power consumption: Should run for 8 hours on four AA batteries.

GPS moving map requirements form

name	GPS moving map
purpose	consumer-grade moving map for driving
inputs	power button, two control buttons
outputs	back-lit LCD 400 X 600
functions	5-receiver GPS; three resolutions; displays current lat/lon
performance	updates screen within 0.25 sec of movement
manufacturing cost	$100 cost-of-goods-sold
power	100 mW
physical size/weight	no more than 2" X 6", 12 oz.

Specification

- A more precise description of the system:
  - should not imply a particular architecture;
  - provides input to the architecture design process.
- May include functional and non-functional elements.
- May be executable or may be in mathematical form for proofs.

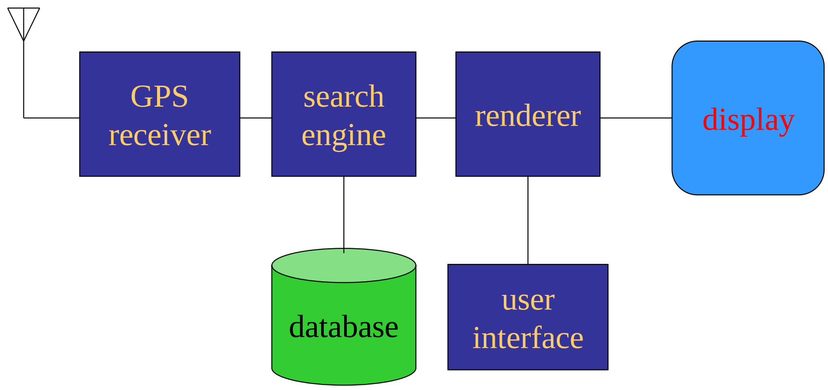
GPS specification

- Should include:
  - What is received from GPS;
  - map data;
  - user interface;
  - operations required to satisfy user requests;
  - background operations needed to keep the system running.

Architecture design

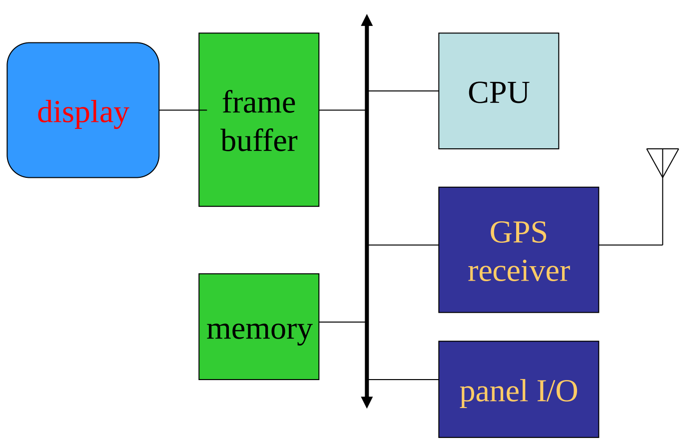
- What major components go satisfying the specification?
- Hardware components:
  - CPUs, peripherals, etc.
- Software components:
  - major programs and their operations.
- Must take into account functional and non-functional specifications.

GPS moving map block diagram



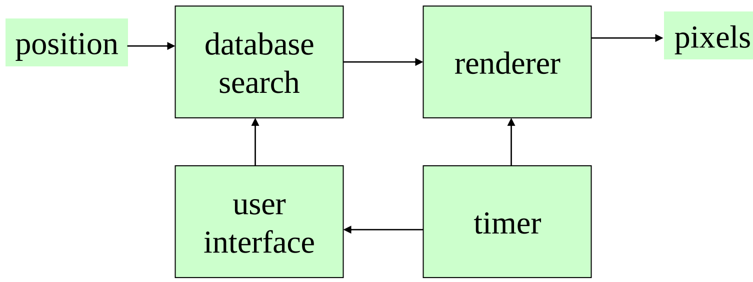
- Operation by SW/HW not yet specified.
- Search database and render(draw) in parallel

GPS moving map hardware architecture



- 2 memories: frame buffer for display, the other for program/data

GPS moving map software architecture



- Add a timer to control when we read the buttons on the user interface and render data onto the screen.

Designing hardware and software components

- Components: Hw (CPU, FPGA, memory,…), Sw
- Must spend time architecting the system before you start coding.
- Some components are ready-made, some can be modified from existing designs, others must be designed from scratch.
- Ensure the system runs properly in real time and the memory space is within allowed.
- Ex. Power consumption is important
  - Memory accesses are major source of power consumption.
  - Memory transactions must be carefully planned to avoid reading the same data several time.

System integration

- Put together the components.
  - 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.
- It is difficult and challenge.
  - Uncover problems
  - Debugging facilities are more limited than what you would find on PC.

Summary

- Embedded computers are all around us.
  - Many systems have complex embedded hardware and software.
- Embedded systems pose many design challenges: design time, deadlines, power, etc.
- Design methodologies help us manage the design process.