Introduction to Embedded Systems

CS/ECE 6780/5780

Al Davis

Today's topics:

- some logistics updates
- ubiquitous diversity of embedded systems
- note – figures obviously copied & w/out credits come from the book

Logistic Changes

- **Lab sections**
  - 1: Wed 1200 – 1500 (William)
  - 2: Wed 1500 – 1800 (Torrey)
  - 3: Fri 0900 – 1200 (Torrey)
  - 4: Fri 1200 – 1500 (William)
- **Mailing lists now work**
  - deadline to sign up stays the same 2359 tomorrow
    - » "git r' done"
  - start using the mailing list after that to find a team mate if needed
- **Next week**
  - teams formed, lab sections designated & lab kits checked out
  - lab 1 will be handed out
- **Week after next**
  - labs start
- **Revised schedule on the website**
  - previous optimistic schedule was clearly bogus

What is an Embedded System?

- Wide variety of interpretations
  - special-purpose computer controlled gizmo
    - » you see the gizmo but not the computer
    - » computer is hidden & not programmed by the user
      - some user configuration is possible
      - but real programming was done by a system specific developer
  - anything that isn't a PC or a server?
    - » reasonable view
    - » note PC's and particularly servers contain ES's
      - thermal controls & cooling are a good example
      - HP's Superdome (class C servers)
        - blades in a rack
        - chip and board thermals reported to rack controller
        - multiple temperature and air flow sensors in the rack
        - servo controlled air flow paddles and numerous fans
        - rack controller
          - controls air flow rate and direction
          - via fan speed and paddle
    - » imagine an ink jet like device in a package
      - sprays coolant to chip hot spots – this stuff works now

ES Functions

- Almost anything you can imagine
  - 5 basic types of functions
    - » capture input signals and process the data
    - » control peripheral circuitry
    - » network with other ES's or computers
      - e.g. sensor arrays in forest canopy
        - NASA study: where the heck is all the extra CO2 coming from?
          - and how much is sequestered in the diminishing forest
        - monitor tremors to predict earthquates
        - monitor tremors to predict volcanic activity
        - monitor your car
      - both fixed and ad-hoc networks
    - » user interfacing
      - something has to read that iPod click wheel
      - or display that new episode of Desperate Housewives
    - » data storage
- Most ES's perform more than one of these functions
  - often with a real-time constraint

Processors

- >99% are placed in ES's
- CNET's Top 10 Must-haves (prices approx.)
  - iPod Touch ($360)
  - Asus 1005HA netbook ($330)
  - Flip UltraHD (mini camcorder - $190)
  - HTC Hero (Android phone - $300)
  - LG enV Touch (phone - $300)
  - Logitech Harmony One (RF remote - $200)
  - Logitech Squeezebox Boom (WiFi radio - $300)
  - Monster Turbine Pro (big bass ear bud - $250)
  - Nintendo Dsi (portable game gizmo - $170)
  - Panasonic TC-P50G10 (HDTV - $1400)
- Notice any trends?

Other Industries

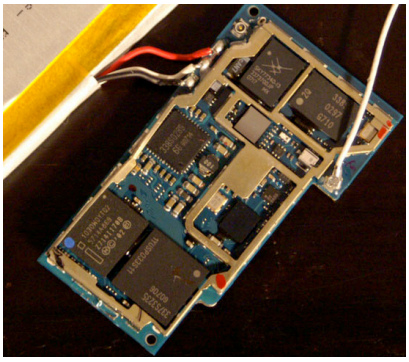
- Automotive
  - air bag controllers, ABS & traction control, engine management
    - » ~50 processors in luxury cars
      - heck my F250 plain jane truck has
        - 2 processors per wheel
        - 2 airbag CPU's
        - 2 ECU's, 2 cruise controllers, + 1 in the radio
        - 9 CPU's in a farmer truck? go figure
      - why? digital control is flexible, reliable, and cheap
    - » CAN bus is on the move
      - every light has a 4-bit controller
- Communications
  - cell phones are very complex ES's
    - » codec's, DSP (iDCT, Rake, Turbo/Viterbi, ...), GPS ...
  - base station has lots of controllers as well

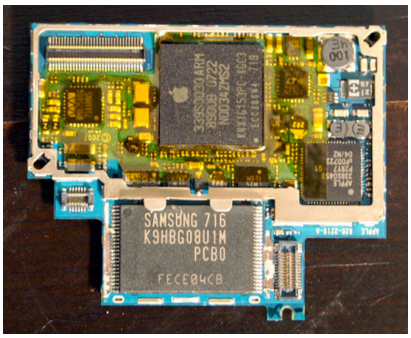
Others

- **Industrial**
  - **robotics**
  - **plant monitoring**
    - » **production optimization**
    - » **critical safety controls (e.g. nuclear, chemical, …)**
- **Commercial**
  - **inventory management – RFID tags on Walmart pallets**
  - **point of sale systems**
- **Medical**
  - **life support monitors**
  - **medical testing**
- **CSI Hoboken**
- **Military**
  - **drones, ATR, G&C, GPS, fighter laser networks,**
  - **too bad this list is endless – then there's DHS (sheesh)**

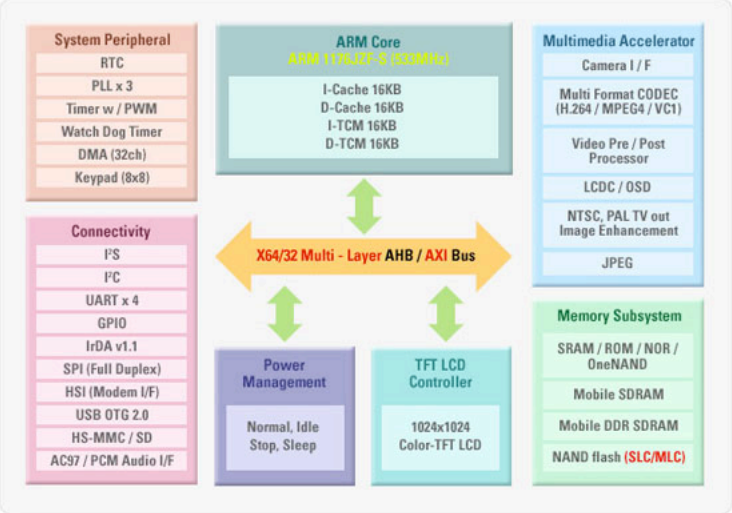
iPhone Dissected

http://www.anandtech.com/printarticle.aspx?i=3026





iPhone (cont'd)



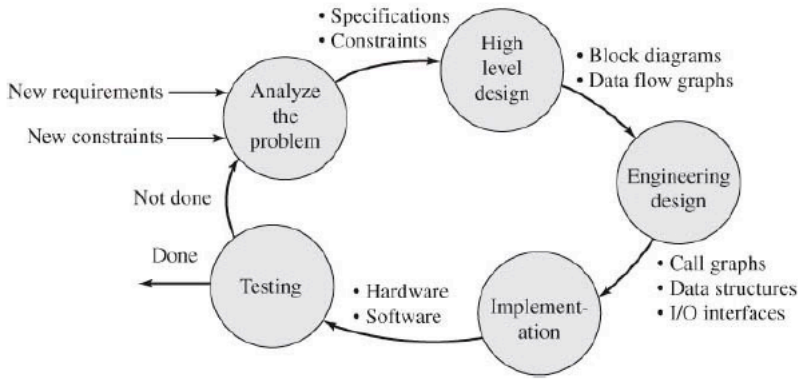
Embedded System Design

- **What makes it unique?**
  - **physically constrained computation**
    - » **timing deadlines & SWAP constraints**
      - **SWAP = size, weight, and power**
  - **traditional abstraction**
    - » **separate software from the HW and environment doesn't work**
  - **HW, SW, & environment**
    - » **tightly intertwined**
    - » **designers must understand the whole space**
- **Great read**
  - **"The Embedded Systems Design Challenge"**
    - » **Henzinger and Sifakis**
  - **continues this discussion**
- **5780 goal**
  - **start this "understanding" process**
  - **5785 continues this process**

Blatant and Shameless Hook

- **Symmetry in CPU's and Jobs**
  - **>99% of CPU's are in ES's**
  - **>99% of the computer jobs may be in ES design as well**
    - » **note the trend**
      - **# of computer companies is not growing**
      - **growth of gizmo companies is rampant**
- **OK**
  - **end of motivational diatribe**

Top-Down Design Process



more details next

Analysis Phase

- Discover the requirements and constraints
  - Requirements
    - » general parameters that the system must satisfy
      - transfer function view
        - – what are the inputs and outputs
  - Specifications
    - » detailed & specific requirements
      - electrical signaling levels
      - mechanical interfaces
      - what functions must the device perform
  - Constraints
    - » e.g. SWAP constraints
    - » environmental issues affecting packaging
      - time for the Schlumberger sour well story
        - – example of failure to understand the environment

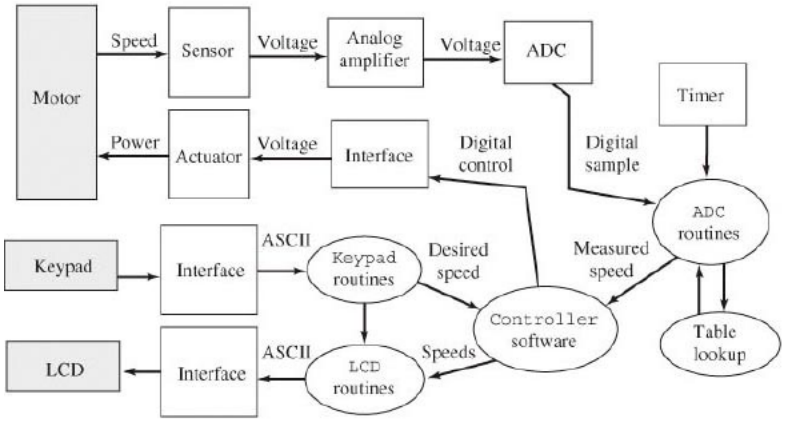
ES Design Metrics

- NRE cost (engineering cost)
- WCT cost (whole cost transfer - $ to make it)
- SWAP
- Performance (accuracy, precision, resolution, response time, bandwidth ...)
- The "bilities"
  - fexible, mantainable, reliable, testable, compatible
- Time
  - to prototype, to market
- Safety
  - Li batteries blow up w/ toxic contents if not charged properly
- User interface
  - look and feel

High Level Design Phase

- **Build conceptual mode of the HW and SW system**
  - **design decomposition**
    - » **define subsystems**
    - » **define interfaces (as in *precisely*)**
- **Estimate**
  - **cost, schedule, performance**
- **Develop**
  - **data flow graph for the entire system**

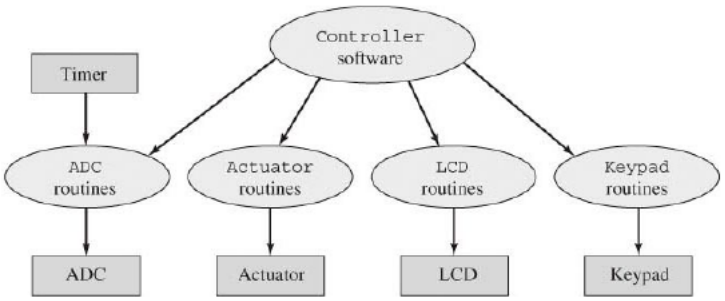
DFG for a Motor Controller



Engineering Design Phase

- **Construct a preliminary design**
  - **includes**
    - » **design hierarchy**
    - » **I/O signals**
    - » **shared data structures**
    - » **overall software scheme**
    - » **power requirements & distribution model**
- **Build mock-ups (not a prototype)**
  - **both mechanical parts and electrical HW**
  - **software**
    - » **both for core function and user interface**
- **Call graphs**
  - **explicit description of how HW & SW modules interact**

Motor Controller Call Graph



Implementation Phase

- **Build the prototype**
  - note some of this could have been started in the design phase
    - » depends on project complexity
- **This can be a very difficult phase**
  - extensive simulation helps in the complex case
    - » HW & SW simulation
    - » cosimulation
  - essential in the very complex case
  - analysis tools are also helpful
    - » mechanical stress testing
    - » power analysis of electrical components
    - » etc.

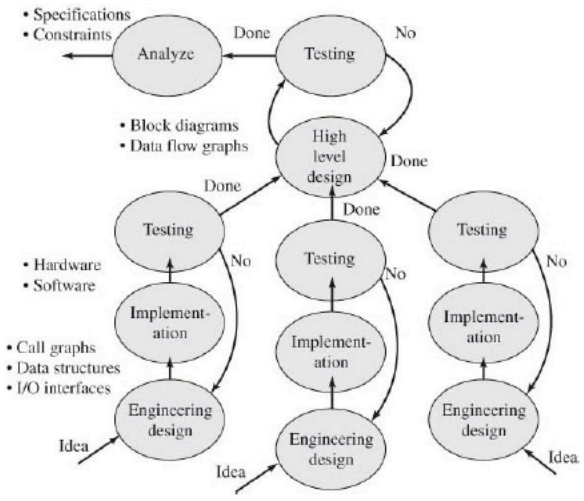
Testing Phase

- **Evaluate how well it works**
  - debug and validate functionality
    - » it MUST work as specified or nothing else matters
  - measure to test compliance of constraints
    - » SWAP
    - » and perhaps some environmental issues
      - final packaging may not yet be in place
      - but vibration, noise sensitive, RF emissions, … come into play
  - optimize various performance parameters
    - » execution speed, accuracy, stability

Maintenance Phase

- **Basic tasks**
  - **correct mistakes**
  - **add new features as required**
  - **further optimization of performance, program size, etc.**
  - **port to new HW or OS platform**
    - » **e.g. new rev of previous system**
    - » **deal with changes in requirements or constraints**
      - **common pain experienced even before 1st release**
      - **culprits: management, market swing, competition**
- **2 possibilities for this phase**
  - **real phase for a released system**
  - **more passes around the loop due to mandated changes**

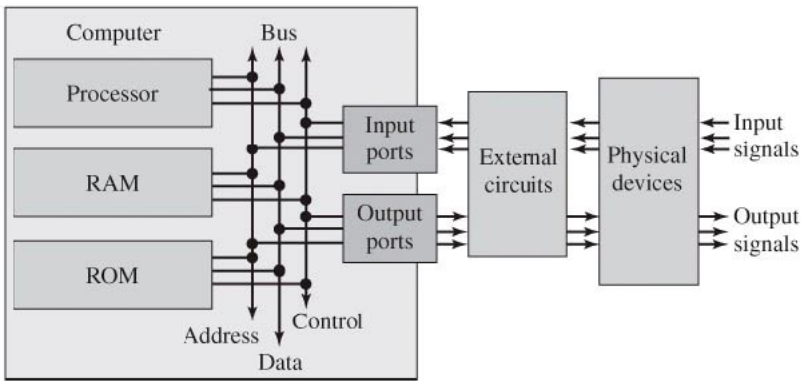
Bottom-Up Design Process



Probable Design Process

- **Neither bottom-up or top-down**
  - **ideas come from both directions**
    - » **bottom: what technology induced opportunities exist**
    - » **top: what the market or a specific customer wants**
  - **meeting in the middle in a balanced way is tricky**
    - » **but it's the game that is played**
- **Hence design is a Yo-Yo**
  - **up-down repeat as necessary**

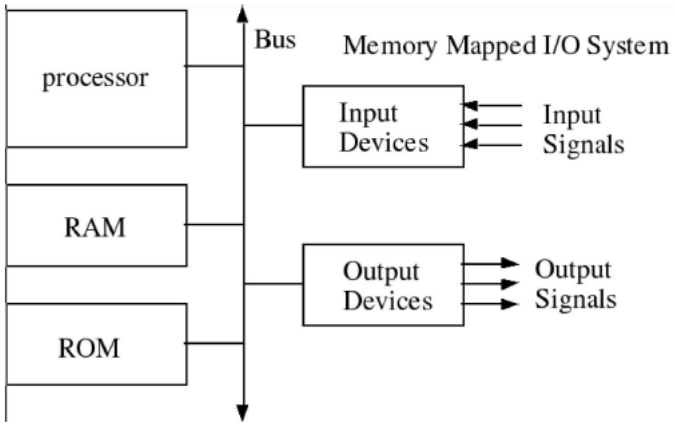
Basic System Components



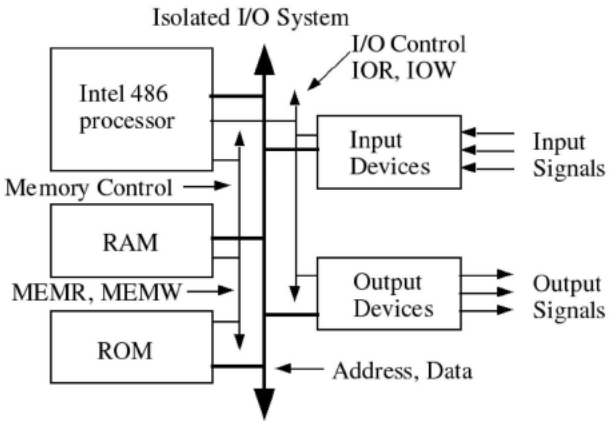
Memory Map

- Processor sources the address & R/W bit
  - but who responds?
- Answer partition the memory space
  - example
    - » $0000-$00FF & Read: read input port values
    - » $0000-$00FF & Write: send data to output ports
    - » $0100-$0FFF – read or write RAM
    - » $1000-$FFFF – read or write ROM
- Actual map is usually device dependent
  - also usually fixed
  - but sometimes user can customize map to some extent

Memory-Mapped System



Isolated I/O System

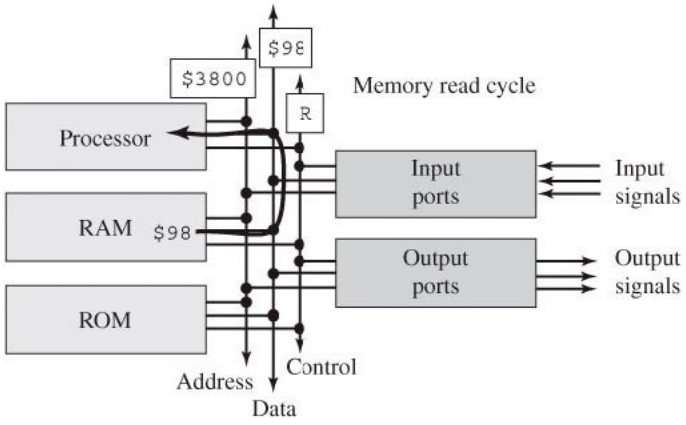


More pins but safer – why?

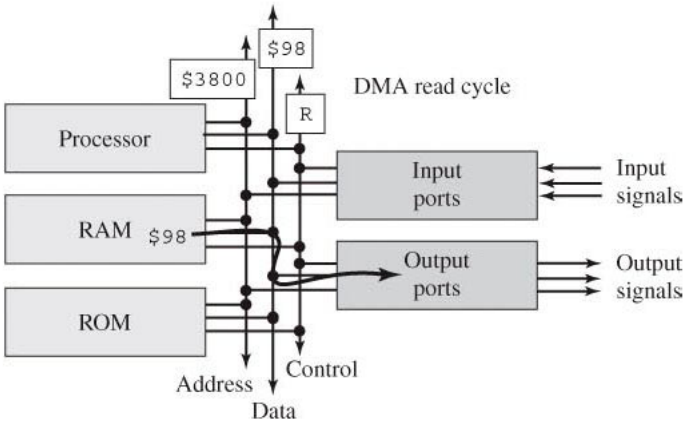
Memory & I/O Essential Differences

- True even when memory is off-chip
- Memory operations can be reordered
  - as long as hazards are avoided
    - » RAW, WAR, WAW
    - » RAR can always be reordered
  - note that the compiler does this
    - » inherent part of optimization
    - » it knows about hazards and how they can be hidden by futzing with pointers
      - a.k.a. obfuscated code
- I/O operations inherently have side effects
  - effectively pseudo-commands
  - must be done in-order and EXACTLY once
    - » so your code will need to tell the compiler somehow to not optimize this for non-I/O isolated systems
      - we'll cover this in more detail later in the course

Memory Read Cycle



DMA Read Cycle



Concluding Remarks

- ES design is different
  - HW/SW/environment intertwine
    - » normal abstract partition wall isn't in place
  - I/O is a much bigger piece of the game
    - » system model is different
    - » CPU is a controller rather than the center of work
  - relatively fixed function
    - » since the peripheral circuitry & components are fixed
    - » control can change but this has a minor effect on system capability
  - reliability is held to much higher standards
    - » inputs say "reactor is going critical"
    - » controller says "can't pull the rods – gotta reboot"