Embedded Systems
In the Real World

Introduction to Embedded Systems
Philip Koopman
January 14, 1999

Further Reading: http://www.ices.cmu.edu/koopman/embedded.html
What is an embedded system?
  • More than just a computer

What makes them different?
  • Real time operation
  • Many sets of constraints on designs

What embedded system designers need to know
  • The big picture
  • Skills required to “play” in this area
WHAT IS AN EMBEDDED SYSTEM?
Definition of an Embedded Computer

- Computer purchased as part of some other piece of equipment
  - Typically dedicated software (may be user-customizable)
  - Often replaces previously electromechanical components
  - Often no “real” keyboard
  - Often limited display or no general-purpose display device

- But, every system is unique -- there are always exceptions
An All-Too-Common View of Computing

- Measured by: Performance
An Advanced Computer Engineer’s View

- Measured by: Performance
  - Compilers matter too…
An Enlightened Computer Engineer’s View

- Measured by: Performance, Cost
  - Compilers & OS matter
An Embedded Computer Designer’s View

- Measured by: Cost, I/O connections, Memory Size, Performance
An Embedded Control System Designer’s View

- Measured by: Cost, Time-to-market, Cost, Functionality, Cost & Cost.
A Customer View

- Reduced Cost
- Increased Functionality
- Improved Performance
- Increased Overall Dependability
  - (Debatable, but can be true)
Three Embedded Examples

◆ **Pocket remote control RF transmitter**
  - 100 KIPS, water/crush-proof, fits in pocket, 5-year battery life
  - Software hand-crafted for small size (less than 1 KB)

◆ **Industrial equipment controller (e.g., elevator; jet engine)**
  - 1-10 MIPS for 1 to 10 CPUs, 1 - 8 MB memory
  - Safety-critical software; real-time control loops

◆ **Military signal processing (e.g., Radar/Sonar)**
  - 1 GFLOPS, 1 GB/sec I/O, 32 MB memory
  - Software hand-crafted for high performance
Small Computers Rule The Marketplace

- ~80 Million PCs vs. ~3 Billion Embedded CPUs Annually
  - Embedded market growing; PC market mostly saturated

$13,490M Total
1994 Worldwide Microcontroller Revenue
($Million U.S.)

4-Bit
$2,200M

8-Bit
$4,520M

16-Bit
$2,910M

32-Bit
$3,840M

64-Bit
$220M

2,683M Total
1994 Worldwide Microcontroller Units
(Million Devices)

4-Bit
1,140M

8-Bit
1,200M

16-Bit
276M

32-Bit
65M

64-Bit
2M

Source: The Information Architects
Approximated from EE Times, March 20, 1995
WHY ARE EMBEDDED SYSTEMS DIFFERENT FROM DESKTOP COMPUTERS?
Four General Embedded System Types

◆ **General Computing**
  - Applications similar to desktop computing, but in an embedded package
  - Video games, set-top boxes, wearable computers, automatic tellers

◆ **Control Systems**
  - Closed-loop feedback control of real-time system
  - Vehicle engines, chemical processes, nuclear power, flight control

◆ **Signal Processing**
  - Computations involving large data streams
  - Radar, Sonar, video compression

◆ **Communication & Networking**
  - Switching and information transmission
  - Telephone system, Internet
Types of Embedded System Functions

- **Control Laws**
  - PID control
  - Fuzzy logic, …

- **Sequencing logic**
  - Finite state machines
  - Switching modes between control laws

- **Signal processing**
  - Multimedia data compression
  - Digital filtering

- **Application-specific interfacing**
  - Buttons, bells, lights,…
  - High-speed I/O

- **Fault response**
  - Detection & reconfiguration
  - Diagnosis
Distinctive Embedded System Attributes

- **Reactive**: computations occur in response to external events
  - Periodic events (*e.g.*, rotating machinery and control loops)
  - Aperiodic events (*e.g.*, button closures)

- **Real Time**: correctness is partially a function of time
  - **Hard real time**
    - Absolute deadline, beyond which answer is useless
    - (May include minimum time as well as maximum time)
  - **Soft real time**
    - Approximate deadline
    - Utility of answer degrades with time difference from deadline
  - In general Real Time != “Real Fast”
Typical Embedded System Constraints

◆ Small Size, Low Weight
  • Hand-held electronics
  • Transportation applications -- weight costs money

◆ Low Power
  • Battery power for 8+ hours (laptops often last only 2 hours)
  • Limited cooling may limit power even if AC power available

◆ Harsh environment
  • Heat, vibration, shock
  • Power fluctuations, RF interference, lightning
  • Water, corrosion, physical abuse

◆ Safety-critical operation
  • Must function correctly
  • Must not function incorrectly

◆ Extreme cost sensitivity
  • $.05 adds up over 1,000,000 units
A SAMPLING OF WHAT EMBEDDED DESIGNERS MUST DEAL WITH
**Embedded System Design World-View**

- **A complex set of tradeoffs**
  - Optimize for more than just speed
  - Consider more than just the computer
  - Take into account more than just initial product design

<table>
<thead>
<tr>
<th>Multi-Objective</th>
<th>Multi-Discipline</th>
<th>Life Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependability</td>
<td>Electronic Hardware</td>
<td>Requirements</td>
</tr>
<tr>
<td>Affordability</td>
<td>Software</td>
<td>Design</td>
</tr>
<tr>
<td>Safety</td>
<td>Mechanical Hardware</td>
<td>Manufacturing</td>
</tr>
<tr>
<td>Security</td>
<td>Control Algorithms</td>
<td>Deployment</td>
</tr>
<tr>
<td>Scalability</td>
<td>Humans</td>
<td>Logistics</td>
</tr>
<tr>
<td>Timeliness</td>
<td>Society/Institutions</td>
<td>Retirement</td>
</tr>
</tbody>
</table>
Mission-Critical Applications Require Robustness

◆ June, 1996 loss of inaugural flight
  • Lost $400 million scientific payload (the rocket was extra)
◆ Efforts to reduce system costs led to the failure
  • Re-use of Inertial Reference System software from Ariane 4
  • Improperly handled exception caused by variable overflow during new flight profile (that wasn’t simulated because of cost/schedule)
    – 64-bit float converted to 16-bit int assumed not to overflow
    – Exception caused dual hardware shutdown (because it was assumed software doesn’t fail)
◆ What really happened here?
  • The narrow view: it was a software bug -- fix it
  • The broad view: the loss was caused by a lack of system robustness in an exceptional (unanticipated) situation
◆ Many embedded systems must be robust
Software Drives Designs

- **Hardware is mostly a recurring cost**
  - Cost proportional to number of units manufactured

- **Software is a “one-time” non-recurring engineering design cost (NRE)**
  - Paid for “only once”
    - But bug fixes may be expensive, or impossible
  - Cost is related to complexity & number of functions
  - Market pressures lead to feature creep
  - *SOFTWARE Is Not FREE!!!!!!*

Source: *Software Requirements: objects, functions, states; Davis, 1993.*
“Let’s use a CAD system to re-synthesize designs for cost optimization”
- Automatically use whatever components are cheap that month
- Would permit quick responses to bids for new variants
- Track record of working fine for PC motherboards

Why wouldn’t it work for an automotive application?
- Embedded system had more analog than digital -- mostly digital synthesis tool
- Cost of re-certification for safety, FCC, warrantee repair rate
- Design optimized for running power, not idle power
  - Car batteries must last a month in a parking lot
- Parts cost didn’t take into account life-cycle concerns
  - Price breaks for large quantities
  - Inventory, spares, end-of-life buy costs
- Tool didn’t put designs on a single sheet of paper
  - Archive system paper-based -- how else do you read 20-year-old files?
Embedded System Designer Skill Set

◆ Appreciation for multi-disciplinary nature of design
  • Both hardware & software skills
  • Understanding of engineering beyond digital logic
  • Ability to take a project from specification through production

◆ Communication & teamwork skills
  • Work with other disciplines, manufacturing, marketing
  • Work with customers to understand the real problem being solved
  • Make a good presentation; even better -- write “trade rag” articles

◆ And, by the way, technical skills too…
  • Low level: Microcontrollers, FPGA/ASIC, assembly language, A/D, D/A
  • High level: Object-oriented Design, C/C++, Real Time Operating Systems
  • Meta level: Creative solutions to highly constrained problems
  • Likely in the future: Unified Modeling Language, embedded networks
  • Uncertain future: Java, Windows CE
REVIEW
Review

◆ What is an embedded system?
  • More than just a computer -- it’s a system

◆ What makes embedded systems different?
  • Many sets of constraints on designs
  • Four general types:
    – General Purpose
    – Control
    – Signal Processing
    – Communications

◆ What embedded system designers need to know
  • Multi-objective: cost, dependability, performance, etc.
  • Multi-discipline: hardware, software, electromechanical, etc.
  • Life cycle: specification, design, prototyping, deployment, support, retirement