CSE 237A
Platforms

Ayse Coskun
Department of Computer Science and Engineering
University of California, San Diego
Embedded System Hardware

- A/D converter
- Sample-and-hold
- Sensors
- Information processing
- Display
- D/A converter
- Actuators
Hardware Platform Architecture
The PC as a Platform

Advantages:
- Cheap and easy to get
- Rich and familiar software environment

Disadvantages:
- Requires a lot of hardware resources
- Not well-adapted to real-time
Host / Target Design

- Use a host system to prepare software for target system:

host system -> serial line -> target system
Host-Based Tools

- Cross compiler:
  - Compiles code on host for target system

- Cross debugger:
  - Displays target state, allows target system to be controlled
Evaluation Boards

- Designed by CPU manufacturer or others
- Includes CPU, memory, some I/O devices
- May include prototyping section
- CPU manufacturer often gives out evaluation board netlist---can be used as starting point for your custom board design
Adding Logic to a Board

- **Programmable logic devices (PLDs)** provide low/medium density logic

- **Field-programmable gate arrays (FPGAs)** provide more logic and multi-level logic

- **Application-specific integrated circuits (ASICs)** are manufactured for a specific purpose
How To Exercise Code

Run on:
- Host system
- Target system
- Instruction-level simulator
- Cycle-Accurate simulator
- Hardware/Software co-simulation environment
Debugging Embedded Systems

- Challenges:
  - Target system may be hard to observe
  - Target may be hard to control
  - May be hard to generate realistic inputs
  - Setup sequence may be complex
Testing and Debugging

- **ISS**
  - Gives us control over time – set breakpoints, look at register values, set values, step-by-step execution, ...
  - But, doesn’t interact with real environment

- **Download to board**
  - Use device programmer
  - Runs in real environment, but not controllable

- **Compromise: Emulator**
  - Runs in real environment, at speed or near
  - Allows you to stop execution, examine CPU state, modify registers.
Debuggers

- A monitor program residing on the target provides basic debugger functions
- Debugger should have a minimal footprint in memory
- User program must be careful not to destroy debugger program, but should be able to recover from some damage
- Breakpoints are very useful
  - Replace the break-pointed instruction with a subroutine call to the monitor program
Breakpoint Handler Actions

- Save registers
- Allow user to examine machine
- Before returning, restore system state
  - Safest way to execute the instruction is to replace it and execute in place
  - Put another breakpoint after the replaced breakpoint to allow restoring the original breakpoint
Platforms Example: Mote

- SW & HW kits

### MoteWorks Architecture

<table>
<thead>
<tr>
<th>Mote Network Tier</th>
<th>Server Tier</th>
<th>Client Tier</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW Development Tools</td>
<td>Gateway Server</td>
<td>Monitoring &amp; Management Tool</td>
</tr>
<tr>
<td>XOtap</td>
<td>XServe</td>
<td>MoteView</td>
</tr>
<tr>
<td>XMesh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TinyOS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Crossbow Hardware Kits

- **Hardware Debug & Programming Tools**
- **Reference Design Documentation**

<table>
<thead>
<tr>
<th>Sensors</th>
<th>Mote Reference Designs</th>
<th>Gateway Boards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature, Humidity, Data Acq</td>
<td>Sensor I/F, Processor Radio Modules</td>
<td>USB, Serial, Ethernet</td>
</tr>
</tbody>
</table>

### Crossbow Devices

- **XMESH**
  - MOTE NETWORK TIER
  - Wireless Packaged Systems
  - Processor Radio Boards
  - Data Acquisition Systems

- **XSERVE**
  - SERVER TIER
  - Network Appliance
  - Ethernet Gateway
  - Serial Gateway

- **MoteView**
  - CLIENT TIER
  - MoteView Software
Mote Apps

- Two prepackaged apps:
  - Environmental monitoring
    - Temperature, humidity, radiation, photosynthetic, barometric pressure
  - Security
    - Magnetometer
    - Passive IR detector
## Mote Specs

<table>
<thead>
<tr>
<th>Mote Hardware Platform</th>
<th>MICAz</th>
<th>MICA2</th>
<th>MICA2DOT</th>
<th>MICA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Models (as of April 2005)</td>
<td>MPR2400</td>
<td>MPR400/410/420</td>
<td>MPR500/510/520</td>
<td>MPR300/310</td>
</tr>
<tr>
<td><strong>MCU</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chip</td>
<td>ATMega128L</td>
<td>ATMega103L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>7.37 MHz, 8 bit</td>
<td>4 MHz, 8 bit</td>
<td>4 MHz, 8 bit</td>
<td></td>
</tr>
<tr>
<td>Program Memory (kB)</td>
<td></td>
<td>128</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRAM (kB)</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td><strong>Sensor Board Interface</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>51 pin</td>
<td>18 pin</td>
<td>51 pin</td>
<td></td>
</tr>
<tr>
<td>10-Bit ADC</td>
<td>7, 0 V to 3 V input</td>
<td>6, 0 V to 3 V input</td>
<td>7, 0 V to 3 V input</td>
<td></td>
</tr>
<tr>
<td>UART</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Other interfaces</td>
<td>DIO, I2C</td>
<td>DIO</td>
<td>DIO, I2C</td>
<td></td>
</tr>
<tr>
<td><strong>RF Transceiver (Radio)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chip</td>
<td>CC2420</td>
<td>CC1000</td>
<td>TR1000</td>
<td></td>
</tr>
<tr>
<td>Radio Frequency (MHz)</td>
<td>2400</td>
<td>315/433/915</td>
<td>433/915</td>
<td></td>
</tr>
<tr>
<td>Max. Data Rate (kbits/sec)</td>
<td>250</td>
<td>38.4</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Antenna Connector</td>
<td>MMCX</td>
<td></td>
<td>PCB solder hole</td>
<td></td>
</tr>
<tr>
<td><strong>Flash Data Logger Memory</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chip</td>
<td></td>
<td>AT45DB014B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection Type</td>
<td></td>
<td>SPI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size (kB)</td>
<td></td>
<td>512</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Default power source</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>AA, 2x</td>
<td>Coin (CR2354)</td>
<td>AA, 2x</td>
<td></td>
</tr>
<tr>
<td>Typical capacity (mA-hr)</td>
<td>2000</td>
<td>560</td>
<td>2000</td>
<td></td>
</tr>
<tr>
<td>3.3 V booster</td>
<td>N/A</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
### Mote Specs (cont’d)

<table>
<thead>
<tr>
<th>Operating Current (mA)</th>
<th>MICAz</th>
<th>MICA2</th>
<th>MICA2DOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATMega128L, full operation</td>
<td>12 (7.37 MHz)</td>
<td>12 (7.37 MHz)</td>
<td>6 (4MHz)</td>
</tr>
<tr>
<td>ATMega128L, sleep</td>
<td>0.010</td>
<td>0.010</td>
<td>0.010</td>
</tr>
<tr>
<td>Radio, receive</td>
<td>19.7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Radio, transmit (1 mW power)</td>
<td>17</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Radio, sleep</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>Serial flash memory, write</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial flash memory, read</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial flash memory, sleep</td>
<td>0.002</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Mote Hardware Platform

<table>
<thead>
<tr>
<th>Mote Hardware Platform</th>
<th>Standard Battery (# required)</th>
<th>Typical Battery Capacity (mA-hr)</th>
<th>Practical Operating Voltage Range (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MICAz</td>
<td>AA (2)</td>
<td>2000, Alkaline</td>
<td>3.6 to 2.7</td>
</tr>
<tr>
<td>MICA2</td>
<td>AA (2)</td>
<td>2000, Alkaline</td>
<td>3.6 to 2.7</td>
</tr>
<tr>
<td>MICA2DOT</td>
<td>Coin</td>
<td>560, Li-Ion</td>
<td>3.6 to 2.7</td>
</tr>
</tbody>
</table>
ATmega128L

- 4 or 8 MHz
- 8 bit
- 128KB Flash
- 4KB EEPROM
- 4KB SRAM
- 133 instructions
  - most single cycle
- 32 gen. regs
- 2 cycle multiplier
- 8 channel, 10 bit ADC; 15 kSamps/s max resolution
Platforms: XScale

- It’s got it all!
- Everything one would need to develop a next generation cell phone or PDA
- Main board block diagram:
Platforms: XScale

- Daughter board block diagram
XScale Processor

- 7 stage pipeline
- 32 bit
- 32 KB instr/data cache; 2kb mini data cache
- 256 Kb SRAM
- Support for various peripherals
- CPU freq: 100-600 MHz; voltage down to 0.85 V
- Found in Blackberry, Treo, IPAQ, etc.
XScale Architecture

### Pipe / Pipestage

<table>
<thead>
<tr>
<th>Main Execution Pipeline</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF1/IF2</td>
<td>Instruction Fetch</td>
</tr>
<tr>
<td>ID</td>
<td>Instruction Decode</td>
</tr>
<tr>
<td>RF</td>
<td>Register File / Operand Shifter</td>
</tr>
<tr>
<td>X1</td>
<td>ALU Execute</td>
</tr>
<tr>
<td>X2</td>
<td>State Execute</td>
</tr>
<tr>
<td>XWB</td>
<td>Write-back</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory Pipeline</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1/D2</td>
<td>Data Cache Access</td>
</tr>
<tr>
<td>DWB</td>
<td>Data cache writeback</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MAC Pipeline</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1-M5</td>
<td>Handles all multiply instructions</td>
</tr>
<tr>
<td>MWB (not shown)</td>
<td>MAC write-back - may occur during M2-M5</td>
</tr>
</tbody>
</table>

### Instruction Cache
- 32K or 16K bytes
- 32 ways
- Lockable by line

### Data Cache
- 32K or 16K bytes
- 32 ways
- wr-back or wr-through
- Hit under miss

### Data RAM
- 28K or 12K bytes
- Re-map of data cache

### Mini-Data Cache
- 2K or 1K bytes
- 2 ways

### Branch Target Buffer
- 128 entries

### IMMU
- 32 entry TLB
- Fully associative
- Lockable by entry

### DMMU
- 32 entry TLB
- Fully Associative
- Lockable by entry

### Fill Buffer
- 4 - 8 entries

### Performance Monitoring

### Power Mgmt Ctrl

### Debug
- Hardware Breakpoints
- Branch History Table

### MAC
- Single Cycle Throughput (16*32)
- 16-bit SIMD
- 40 bit Accumulator

### JTAG

### Memory pipeline

D1  D2  DWB
Compare Power / Performance

- **XScale**
  - **Active:**
    - 13 MHz => 44.2 mW; 624 MHz => 925 mW
  - **Idle:**
    - 13 MHz => 8.2 mW; 624 MHz => 260 mW
  - **Deep sleep:** 0.1 mW

- **ATmega**
  - **Active:** 4 MHz => 17 mW
  - **Idle:** 4 MHz => 12 mW
  - **Sleep:** 45 uW
Power Management
Energy Efficiency

- Energy-efficient computing is needed by:
  - portable systems
    - increase battery lifetime
    - optimize thermal design -> form factor
  - non-portable systems
    - minimize cost
    - environmental concerns

- Electronic system design
  - Hardware: processing, storage, communication
  - Software: operating systems, applications

- Electronic system utilization
  - Runtime control and management
  - e.g. Dynamic Power Management (sleep states) & Dynamic Voltage Scaling (lower CPU frequency/voltage)
Portable Computers

- Most energy consumed in: display, hard disks, WLAN

**Hard Disk**

- Active power: 0.95 - 2.5 W
- Idle power: 0.95 W
- Sleep: 0.13 W
- Sleep time: 0.67 s
- Wake-up time: 1.6 s

**WLAN**

- Transmission: 1.65 W
- Receiving: 1.4 W
- Doze: 0.045 W
- Down time: 62 ms
- Wake-up time: 34 ms
Power and Energy Relationship

\[ E = \int P \, dt \]
Low Power vs. Low Energy

- Minimizing the **power consumption** is important for
  - the design of the power supply
  - the design of voltage regulators
  - the dimensioning of interconnect
  - short term cooling

- Minimizing the **energy consumption** is important due to
  - restricted availability of energy (mobile systems)
    - limited battery capacities (only slowly improving)
    - very high costs of energy (solar panels, in space)
  - cooling
    - high costs
    - limited space
  - dependability
  - long lifetimes, low temperatures
Power density continues to get worse

Surpassed hot-plate power density in 0.5\(\mu\)
Not too long to reach nuclear reactor
Consider CPU & System Power

Mobile PC
Thermal Design (TDP) System Power

- 600/500 MHz uP: 37%
- LCD 10": 19%
- HDD: 9%
- Memory+Graphics: 12%
- Power Supply: 10%
- Other: 13%

Note: Based on Actual Measurements

CPU Dominates Thermal Design Power

Mobile PC
Average System Power

- 600/500 MHz uP: 13%
- LCD 10": 30%
- Memory+Graphics: 15%
- HDD: 19%
- Power Supply: 10%
- Other: 13%

Multiple Platform Components Comprise Average Power

[Courtesy: N. Dutt; Source: V. Tiwari]
Power Manageable Components

- Components with several internal states
  - Corresponding to power and service levels
- Abstracted as **power state machines**
  - State diagram with:
    - Power and service annotation on states
    - Power and delay annotation on edges

**Example: SA-1100**

- **RUN**: operational
- **IDLE**: a sw routine may stop the CPU when not in use, while monitoring interrupts
- **SLEEP**: Shutdown of on-chip activity
Example: Hard Disk Drive

Fujitsu MHF 2043 AT

- **Working**: 2.2 W (spinning + IO)
- **Idle**: 0.95 W (spinning)
- **Sleeping**: 0.13 W (stop spinning)

- **Spin up**: 4.4J, 1.6 sec
- **IO complete**: 0.36 J, 0.67 sec
- **Shut down**: 0.36 J, 0.67 sec
- **Read / write**: 4.4 J, 1.6 sec

Example: Hard Disk Drive

Fujitsu MHF 2043 AT

- **Working**: 2.2 W (spinning + IO)
- **Idle**: 0.95 W (spinning)
- **Sleeping**: 0.13 W (stop spinning)

- **Spin up**: 4.4J, 1.6 sec
- **IO complete**: 0.36 J, 0.67 sec
- **Shut down**: 0.36 J, 0.67 sec
- **Read / write**: 4.4 J, 1.6 sec
The Applicability of DPM

- State transition power ($P_{tr}$) and delay ($T_{tr}$)

- If $T_{tr} = 0$, $P_{tr} = 0$ the policy is trivial
  - Stop a component when it is not needed

- If $T_{tr} \neq 0$ or $P_{tr} \neq 0$ (always…)
  - Shutdown only when idleness is long enough to amortize the cost
  - What if $T$ and $P$ fluctuate?
Workload and System Representation

- **Workload**: Requests
  - **Requests**
  - **Busy**
  - **Idle**
  - **Working**
  - **Sleeping**
  - **Power State**
  - **Power**

- **System**: Time
  - **Time**
  - **Shutdown Delay**
  - **Wakeup Delay**

**Diagram Notes**:
- **Shut Down**
- **Wake Up**
Waking Up Hard Disk

Measurements done on Fujitsu MHF 2043 AT hard disk
System Break-Even Time: $T_{BE}$

Minimum idle time for amortizing the cost of component shutdown

$$T_{BE} = T_{tr} + T_{tr} \frac{P_{tr} - P_{on}}{P_{on} - P_{off}}$$

Transition delay ($T_{tr}$)

Transition power ($P_{tr}$)

Sleep power ($P_{off}$)
Fixed Time-Out

◆ Simple policy
  ◆ If $T_{idle} > T_{TO}$ go to SLEEP
  ◆ Stay in sleep until workload $!= 0$

◆ Rationale
  ◆ When $T_{idle} > T_{TO}$ it is likely that: $T_{idle} > T_{TO} + T_{BE}$

◆ Choice of $T_{TO}$ is critical
  ◆ Large is safe, but it could be useless
  ◆ Too small is highly undesirable

◆ Limitations
  ◆ Performance penalty for wake-up is paid after every shutdown
  ◆ Power is wasted during $T_{TO}$
OS Based Power Management

- In systems with an operating system (OS)
  - The OS knows of tasks running and waiting
  - The OS should perform the DPM decisions

- Advanced Configuration and Power Interface (ACPI)
  - [Intel, Microsoft, Toshiba]
  - Open standard for design of OS-based power management
**DPM and Operating Systems**

- **Application**
  - should not directly control hardware power
  - no power management in legacy programs

- **Scheduler**
  - selects processes and affects idle periods

- **Process manager**
  - knows multiple requesters
  - can estimate idle periods more accurately

- **Driver**
  - detects busy and idle periods

- **Device**
  - consumes power
  - should provide mechanism, not policy

<table>
<thead>
<tr>
<th>application programs</th>
<th>operating system</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduler</td>
<td>process manager</td>
</tr>
<tr>
<td>device driver</td>
<td>hardware devices</td>
</tr>
</tbody>
</table>
Low-Power Scheduling

Tasks specify
- device requirements
- timing requirements

Operating system
1. Groups tasks with same device requirements
2. Execute tasks in groups
3. Wake up devices in advance to meet timing constraints

Diagram of task execution and device scheduling.
Dynamic Voltage Scaling (DVS)

Power consumption of CMOS circuits (ignoring leakage):

\[ P = \alpha \ C_L \ V^2_{dd} \ f \]

- \( \alpha \): switching activity
- \( C_L \): load capacitance
- \( V_{dd} \): supply voltage
- \( f \): clock frequency

Delay for CMOS circuits:

\[ \tau = k \ C_L \ \frac{V_{dd}}{(V_{dd} - V_t)^2} \]

- \( V_t \): threshold voltage
- (\( V_t \) substantially < than \( V_{dd} \))
Voltage scaling: Example

Maximum Clock Frequency

Energy Consumption

[Courtesy, Yasuura, 2000]
Variable-voltage/frequency

POWER-PERFORMANCE COMPARISON

Intel® StrongARM® Technology

Intel® XScale™ Microarchitecture

![Bar Chart]

- MIPS
- Power Consumption (Watts)

- 233 MHz @2.0V
- 175 MHz @1.5V
- 150 MHz @0.75V
- 400 MHz @1.0V
- 600 MHz @1.3V
- 800 MHz @1.6V
- 1 GHz @1.8V

From Intel's Web Site
Mini Project: Portable Multimedia Player on Intel XScale
Project Goal

- Design an efficient & low power media player for an XScale platform
  - Real time decoding of audio
  - Energy efficiency – use DVS!

- Steps:
  - Install OS onto Intel’s XScale DVK (Linux)
  - Set up cross compiling & debugging environments
  - Get media player to run in real time
  - Minimize CPU power with real time playback
Power Management on XScale

- XScale 27x
  - 150 us for f/V change
  - 0.5 ms to sleep state

<table>
<thead>
<tr>
<th>State</th>
<th>Active (mW)</th>
<th>Idle (mW)</th>
<th>Freq (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>925</td>
<td>260</td>
<td>624</td>
</tr>
<tr>
<td>P2</td>
<td>747</td>
<td>222</td>
<td>520</td>
</tr>
<tr>
<td>P3</td>
<td>279</td>
<td>129</td>
<td>208</td>
</tr>
<tr>
<td>P4</td>
<td>116</td>
<td>64</td>
<td>104</td>
</tr>
<tr>
<td>Psleep</td>
<td>-</td>
<td>0.163</td>
<td>0</td>
</tr>
</tbody>
</table>
Reducing Power Consumption

- Implement a **dynamic voltage frequency scaling policy**
  - Check system status
    - Active / Idle, cpu utilization, etc.
  - Adjust V / f setting accordingly

- Compiling the applications with different optimizations and libraries can reduce power consumption further
  - Try for mplayer
DVS Policies

- Example:
  - The application will not fully utilize the CPU
  - Check utilization
    - e.g. with “top” command
  - Reduce V / f setting to take advantage of low CPU utilization
  - How can you estimate the utilization in the next interval?
How To Test Your Policies

- Run the provided real time applications (without DVS) on the PC and record the execution time
- Run the apps on the platform
- If they’re slower → not real time!
Sources and References

- Nikil Dutt @ UCI