Low Power and Low Energy Software

Embedded Systems Design Course
EDAN15

Software P/E Estimation

- Instruction profiling (fast but rather inaccurate)
  - use average power/energy for each instruction (or pair) pre-measured on a real processor
  - Estimated energy of a sequence of code = \sum energies for all instructions (pairs)
- Detailed modeling (accurate but very slow)
  - Accurately model the processor using low abstraction level descriptions (gate, RT)
  - Simulate the hardware (+RAM with the program) and get power/energy figures

Methods Overview

- Architecture Selection
  - What hardware resources we use?
  - How do we connect them?
  - Which software is executed on which hardware?
- Offline Decisions
  - Taken before the system becomes operational (compilation, scheduling type, OS,...)
- Run-time Decisions
  - Adaptive, depending on the input data and load (power management)

Architecture Selection: Which Processors?

- Power-Performance tradeoff (MIPS/Watt Efficiency)
- Power Modes, Dynamic Voltage Supply

Architecture Selection: Work Load Distribution?

Observations:
- Off-chip communications usually take a lot of energy -> SoC
- More parallelism allows for slower but energy efficient processors

Example: 8 tasks, 9 comm +3 proc types, 1 bus

Offline Decisions (I)

- Compilation for low energy focuses on:
  - Algorithm selection
  - Instruction set selection (+Thumb)
  - Instruction reordering/selection & Register renaming
    - Reduces the switching on the data bus
  - Data to memory mapping
    - Reduces the switching on the address bus
    - Increases cache efficiency
Offline Decisions (II)

- Voltage Selection (i.e. Intra-Task worst case path)

```c
if (a > d) {
    a = b * c;
    d = a/2;
    a -= sqrt(d);
} else {
    a += d/2;
}
```

- after each decision that may reduce the worst case path, new code for adjusting speed and voltage is inserted.

Run-Time Decisions (I)

- Power Management:
  - power down resources when idle
  - use lazy I/O:
    - only power up peripherals if the buffers are full/write or empty/read
  - predictive power down/up for improved performance
  - correlate the resource power modes with the slowest to power up
  - usually OS-level

Run-Time Decisions (II)

- Voltage/Speed Scheduling (only on DVS processors)
  - integrated with task scheduling
  - correlate processor speed with its utilization
  - use minimal speeds to meet a deadline
  - pass on unused processor time from tasks finishing early

! Even Hard Real-Time systems can be energy efficient with proper voltage scheduling.

Voltage Selection Code Example

Intel 80200EVB [XScale]

```assembly
ldr r1, MHZ333
mcr p14,0,r1,c6,c0,0
\; 400 to 333 MHz PLL relock
```

Power Management Example: ACPI

Compaq, Intel, Microsoft, Phoenix, Toshiba

Advanced Configuration and Power Interface (ACPI) 2.0 specification

Voltage/Speed Scheduling Example

```
```

```
```
Summary

- Power and energy efficiency are important design requirements today.
- Can be addressed at all times throughout the design process (both in Hardware & Software).
- Processors with Power Modes and Dynamic Voltage Supply are a must in low-energy embedded designs.
  (most do support these today!)