Low-Power and Temperature-Aware Compilation for Embedded Processors

José L. Ayala
Politecnica University of Madrid
jayala@die.upm.es

Outline

- Motivation
- Goals
- Low-Power Compilation
- Temperature-Aware Compilation
- Ongoing and Future Work

Motivation

- Continuing advances in semiconductor technology → performance gains (clock rate and ILP)
- Significant increase in power consumption (dynamic and static)
- Technology scales → future power density 200 W/cm^2: early consideration of thermal effects are needed

Goals

- Extensive efforts on dynamic techniques: static approaches are needed
- Early thermal characterization with a selectable granularity
- Thermal and power management incorporated in the compiler code

Low-Power Compilation

- Power-aware compilation for embedded single-core processors
  - Modified register assignment: power reduction in the register file
  - Register file organized in banks: simple decoding logic
  - Register assignment improves register locality: registers assigned from same bank

Low-Power Compilation

- Power-aware compilation for embedded single-core processors
  - Voltage scaling technique applied to unused banks
  - Compiler modifications in gcc compiler
  - Energy savings up to 75% without performance penalty

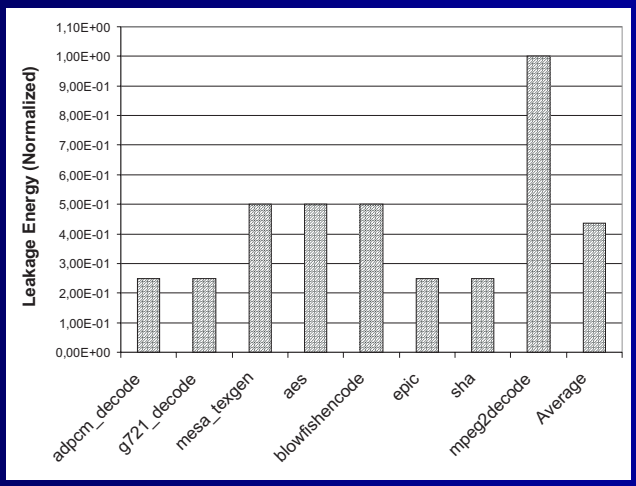
Low-Power Compilation

- Power-aware compilation for VLIW-based systems
  - Power-aware register assignment targeting VLIW-based systems
  - Similar approach than the applied to embedded processors
  - Compiler modifications in Trimaran compiler

Low-Power Compilation

- Power-aware compilation for VLIW-based systems
  - Simulation framework: CRISP (by IMEC)
  - Energy savings up to 60% without performance penalty
  - Modified compiler helps on simplifying over-sized register file in compilation time

Low-Power Compilation



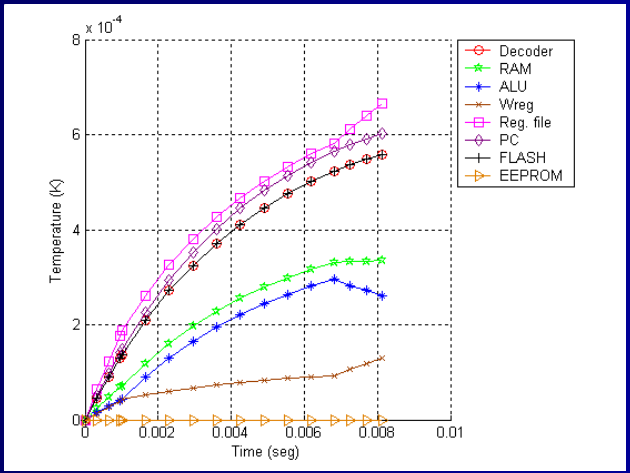
Temperature-Aware Compilation

- Two different goals:
  - Decrease total temperature in the chip: effect on propagation delays, signal integrity and power consumption
  - Avoid temperature gradients: effect on electromigration and chip damage

Temperature-Aware Compilation

- First step: thermal modeling
  - Thermal modeling based on power consumption measures: bidimensional RC thermal model through layout area estimations (*black box* model)
  - Thermal characterization of hardware modules in the chip area

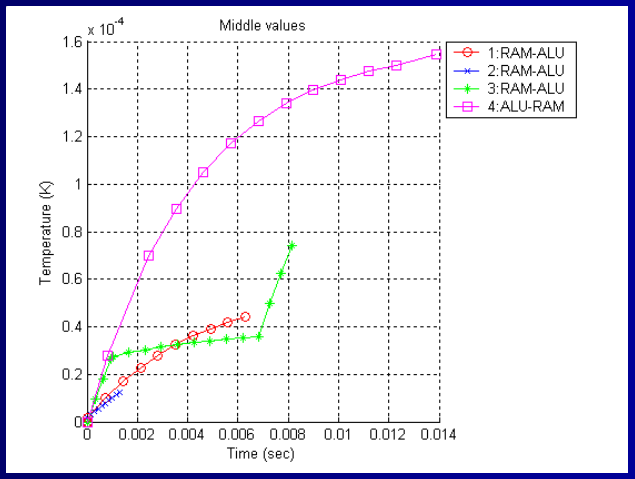
Temperature-Aware Compilation



Temperature-Aware Compilation

- Second step: thermal analysis of source-level transformations
  - Analysis of temperature gradients

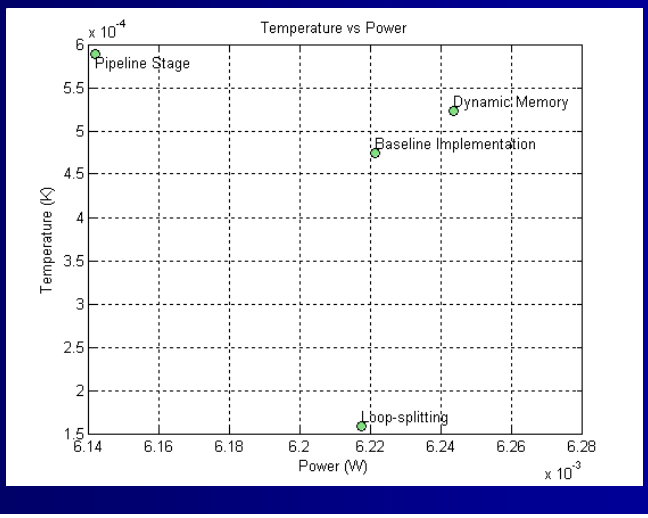
Temperature-Aware Compilation



Temperature-Aware Compilation

- Second step: thermal analysis of source-level transformations
  - Analysis of temperature gradients
  - Analysis of global temperature

Temperature-Aware Compilation



Temperature-Aware Compilation

- Third step: temperature/power-aware compiler

Ongoing and Future Work

- Compiler-driven reconfigurable logic for low-power consumption: improves register assignment
- Implementation over a real platform (CoolFlux, Lisatek, ACE/COSY...)
- Extension of the number of analyzed code transformations with impact in the thermal behavior

Ongoing and Future Work

- Integration of low-power optimizations and temperature-aware transformations in the compiler code

Thanks!

Questions?