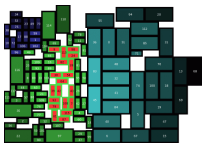
Design and Synthesis of Embedded Systems

Robert Dick

http://ziyang.eecs.northwestern.edu/~dickrp
Department of Electrical Engineering and Computer Science
Northwestern University





Embedded system definition

Embedded system: A computer within a host device, when the host device itself is not generally considered to be a computer.

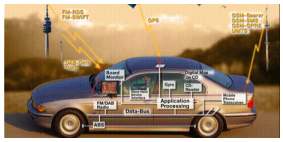
Not a general-purpose desktop computer.

In many applications, well-designed, correctly functioning embedded systems are almost invisible to their users.

Embedded systems examples


Medical devices


Sensor networks


Automobiles


Smartphones

Embedded system market size

Dominates general-purpose computing market in volume

Similar in monetary size to general-purpose computing market

Growing at 15% per year, 10% for general-purpose computing

Embedded system requirements

Hard real-time: Deadlines must not be violated

Wireless: Effects of the communication medium important

Reliable: Better crash desktops than cars

First time correct: Field repairs difficult

Rapidly implemented: IP use, HW–SW co-design

Low price: Fierce competition between many companies

High-performance: Massively parallel, using ASICs

Low power: Battery life and cooling costs

CMP Media LLC survey

Conflicting expectations make design difficult and unpredictable

- 1,100 embedded system developers
- Majority of projects were running late
  - Four-month delay normal
- Majority had lower performance than predicted
  - 50% expected and planned performance normal

Design process unpredictability due to manual, ad-hoc design

Our research goals

Develop better embedded system design ideas

Automate embedded system design process

Problem background

RAM quantity limits application functionality

RAM price dropping but usage growing faster

Secure Internet access, email, music, and games

How much RAM?

- Functionality
- Cost
- Power consumption
- Size

Ideal hardware–software design process

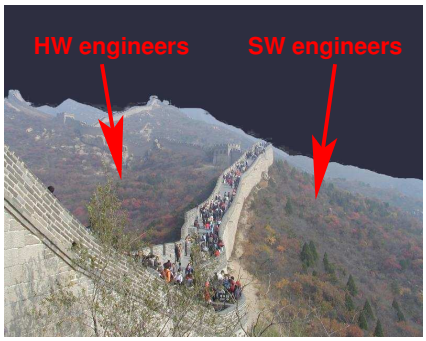
Ideal case

Hardware and software engineers collaborate on system-level design from start to finish

We teach the advantages of this in our classes

It doesn't always happen

Real hardware–software design process



Options

Option 1: Add more memory

Implications: Hardware redesign, miss shipping target, get fired

Option 2: Rip out memory-hungry application features

Implications: Lose market to competitors, fail to recoup design and production costs, get fired

Option 3: Make it seem as if memory increased without changing hardware, without changing applications, and without performance or power consumption penalties

Nobody knew how to do this

Goals

Allow application RAM requirements to overrun initial estimates even after hardware design

Reduce physical RAM, negligible performance and energy cost

Improve functionality or performance with same physical RAM

Past work

HW RAM compression: Tremaine 2001, Benini 2002, Moore 2003

- Hardware (de)compression unit between cache and RAM
- Hardware redesign and application-specific compression hardware
- Past work claimed application-specific hardware essential to keep power and performance overhead low

Design principles

Page selection

Scheduling compression and decompression

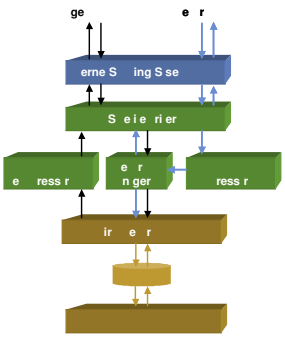
Organizing compressed and uncompressed regions

Dynamically adjust compressed region size

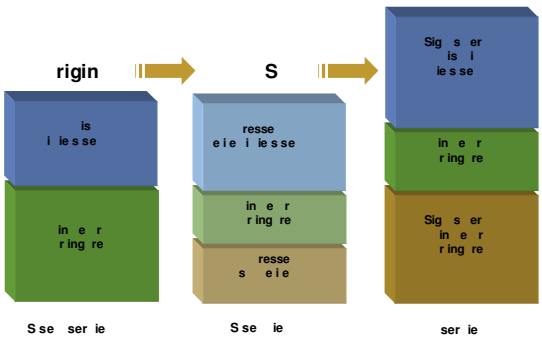
Compression scheme

- High performance
- Energy efficient
- Good compression ratio
- Low memory requirement

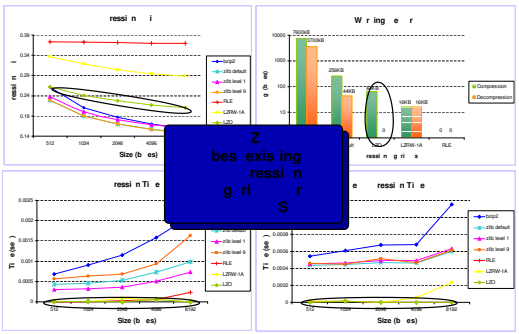
CRAMES design



System configuration



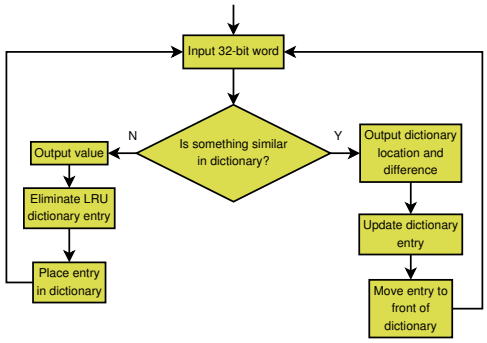
Compression algorithm



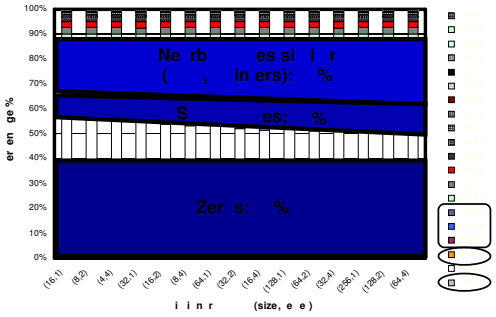
Weak link: Compression algorithm

- LZO average performance penalty 9.5% when RAM reduced to 40%
- Developed simulation environment to permit profiling
- Compression and decompression were taking most time
- Needed a better compression algorithm

Pattern-based partial dictionary match coding



Data regularity



Pattern-based partial match

Algorithm
- Consider each 32-bit word as an input
- Allow partial dictionary match
- Use most frequent patterns based on statistical analysis

Optimizations
- Two-way associative LRU 16-entry hash-mapped dictionary
- Optimized coding scheme
- Early termination for uncompressable data
- Fine-grained operation parallelization

Experimental setup

Sharp Zaurus SL-5600 PDA
- Intel XScale PXA250
- 32 MB flash memory
- 32 MB RAM
- Embedix (Linux 2.4.18 kernel)
- Qt/Qtopia PDA edition

Benchmarks

PBPM twice as fast as LZO

Impact of using CRAMES to reduce physical RAM

Results for
- ADPCM: Speech compression application from MediaBench
- JPEG: Image encoding application from MediaBench
- MPEG2: Video CODEC application from MediaBench
- Straight-forward matrix multiplication
  - Intentionally difficult for CRAMES

Also tested on
- 10 GUI applications that came with Qtopia
- Next-generation cellphone prototype

Results

Reduced RAM from 20 MB to 8 MB

Base case: 20 MB RAM, no compression

Without CRAMES
- All suffered significant performance penalties
- Matrix multiplication cannot execute

With CRAMES
- LZO average case 9% overhead, worst case 29%
- PBPM average case 2.5% overhead, worst case 9%

Also works on arbitrary in-RAM filesystems

Status

Patent pending: Northwestern University and NEC

Millions of cellphones using CRAMES ship in Fall 2007

However, only shipping in Japanese market

Motorola interested in licensing

- NEC worked closely with us and provided support
- Open to licensing, but only later
- After they are first to market with the technology

Until then...

Until then, NEC gave us these T-shirts

Wireless sensor networks

Self-organized wireless networks of sensors

Extremely tight resource constraints

- Limited performance processor
- Tight memory constraints, e.g., 4 KB
  - Have solution based on compiler technology
- Energy constraints

Event-driven applications

Events occur at unpredictable times

E.g., structural integrity monitoring

Should not be ignored

- Existing sensor network nodes only poll
- Some have proposed SW solutions, e.g., Zheng 2003
  - Will either miss events or waste power
- Designed hardware solution

Lucid dreaming

16 µW event detector power consumption

Extends battery life from weeks to many months

Low-power motivation

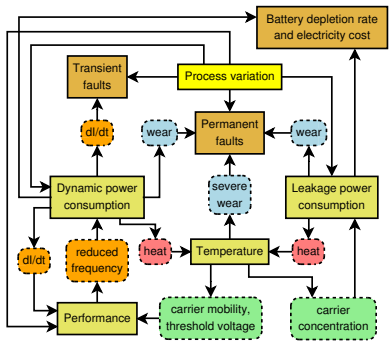
High power consumption results in

- Expensive, bulky packaging
- Limited performance
- Short battery life
- Reduced reliability

High-level trade-offs among

Power, speed, price, area, and temperature

Power and its associated evils



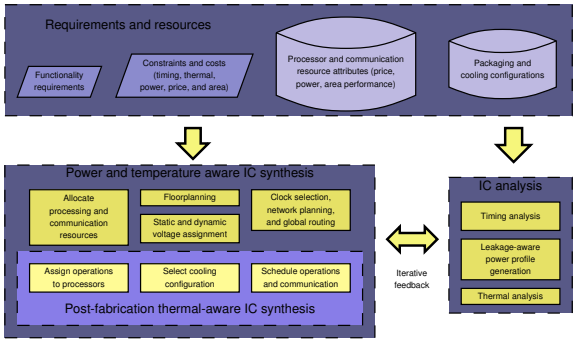
Synthesis motivation and definition

VLSI ICs among most complex systems designed by humans

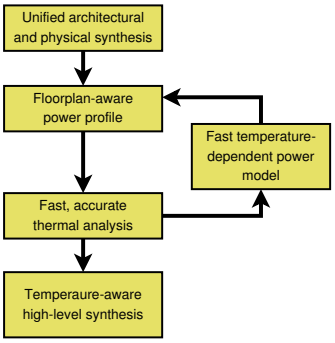
- Automation is essential
- Manual design no longer possible

Synthesis is the use of algorithms for automatic design

Power and temperature aware IC synthesis components



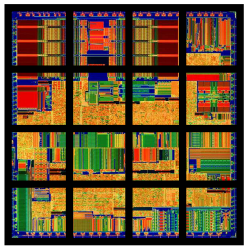
Example temperature-aware synthesis flow

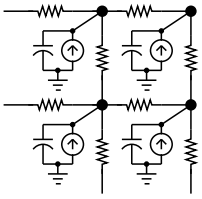


Thermal analysis requirements

- Non-linear impact on reliability and other design characteristics
- Must be accurate
- Speed necessary for thorough design exploration
- Architectural design or synthesis require many thousands of invocations
- Must be extremely fast

RC model





Partition into 3-D elements (diagram 2-D for simplicity)

Thermal resistance $\leftrightarrow$ Resistance

Heat flow $\leftrightarrow$ Current

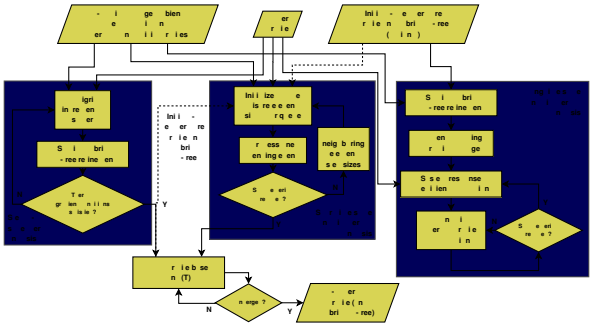
For dynamic: Heat capacity $\leftrightarrow$ Capacitance

Problem definition

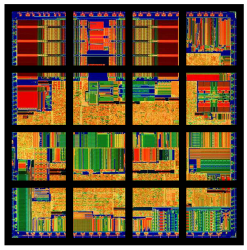
$$\mathbf{C}\frac{d\mathbf{T}(t)}{dt} = \mathbf{A}\mathbf{T}(t) + \mathbf{P}U(t)$$

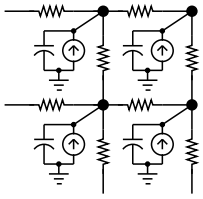
- $\mathbf{A}$ is the thermal conductivity matrix
- Steady-state: Initial temperature and $\mathbf{C}$ unnecessary
- Dynamic: Transient temperature analysis, must also consider heat capacity

Thermal analysis infrastructure overview



RC model





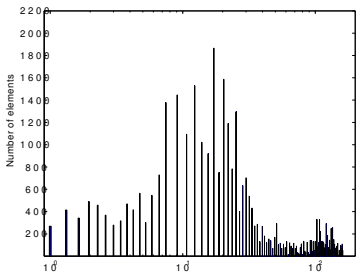
Partition into 3-D elements (diagram 2-D for simplicity)

Thermal resistance $\leftrightarrow$ Resistance

Heat flow $\leftrightarrow$ Current

For dynamic: Heat capacity $\leftrightarrow$ Capacitance

Histogram of maximum safe step sizes



Asynchronous temporal adaptation can improve performance w.o. loss of accuracy

Asynchronous time marching

Allow step sizes to differ in space and time

This eliminates local time synchronization

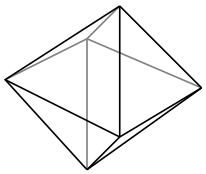
How to handle steps when neighbors at different time?

Asynchronous elements

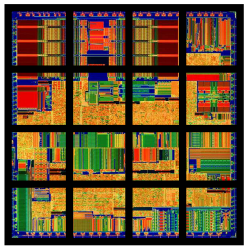
- Local time estimation expensive for higher-order methods
- For each element, compute partial results based on n neighbors

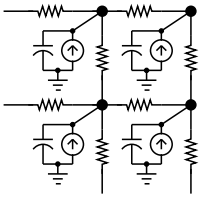
$$n = (4d^3/3 + 2d^2 + 8d/3)$$

- Discretized octahedron
- $|E|$ is the number of elements
- d is the transitive neighbor depth



RC model





Partition into 3-D elements (diagram 2-D for simplicity)

Thermal resistance $\leftrightarrow$ Resistance

Heat flow $\leftrightarrow$ Current

For dynamic: Heat capacity $\leftrightarrow$ Capacitance

Example asynchronous method

$$0 = \sum_{i=1}^{6} \frac{T(t) - T_i \cdot u(t)}{R_i} + C\frac{dT}{dt} - P \cdot u(t)$$

By Laplace transform, linearity theorem, and inverse Laplace transform.

$$\frac{dT}{dt} = \left(\frac{\sum_{i=1}^{6} T_i/R_i + P - T(0^-) \cdot \sum_{i=1}^{6} 1/R_i}{C} \right) \cdot e^{-t/C \sum_{i=1}^{6} 1/R_i}$$

Allows computation of temperature after time step.

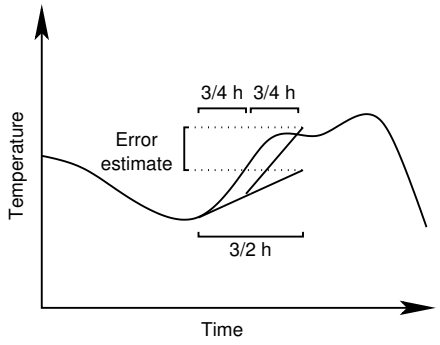
Asynchronous temporal adaptation

- Neighbors at different times
- Extrapolate neighbor temperatures to take step
- Adapt step size by taking two $3/4\ h$, one $3/2\ h$ steps and comparing

$$s_i(t_i) = u \cdot \sqrt[v]{\frac{y}{\left| \frac{dT_i}{dt}(t_i) \cdot \frac{3}{2} \cdot h_i - \frac{3}{4} \cdot h_i \left(\frac{dT_i}{dt}(t_i) + \frac{dT_i}{dt}(t_i + \frac{3}{4} \cdot h_i) \right) \right|}}$$

where v is the order of the method in use

Step size adaptation



Asynchronous time marching validation

Problem	ISAC				GARK4	
	CPU time (s)	Speedup (×)	Mem. (KB)	Error (%)	CPU time (s)	Mem. (KB)
chemical	1.35	1354	463.47	0.13	1827.41	4,506
dct_wang	0.39	1457	312.64	0.09	568.22	4,506
dct_dif	0.40	1807	332.91	0.05	722.64	4,506
dct_lee	0.85	1071	439.22	0.04	910.88	4,506
elliptic	2.24	1361	412.23	0.02	3042.61	4,506
iir77	0.86	1521	803.09	0.08	1305.25	4,506
jcb_sm	0.58	1890	357.30	0.11	1092.98	4,506
mac	1.65	1105	403.47	0.45	1817.71	4,506
paulin	0.77	1439	354.28	0.18	1111.68	4,506
pr2	1.06	1831	489.36	0.35	1932.95	4,506

Optimal temperature-aware real-time scheduling

Developed MILP formulation of temperature-aware real-time assignment and scheduling problem

Minimize peak temperature under hard time constraints

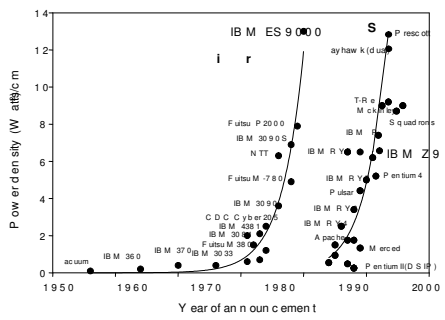
CPLEX can optimally solve for ICs with 3×3 processor cores

Compared to optimal energy consumption minimization

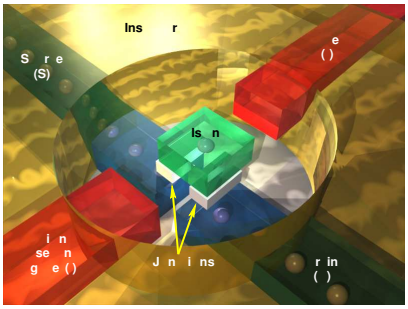
- Peak temperature reduction of 8.7 °C, on average
- Peak temperature reduction up to 24.7 °C

Developed fast heuristic that deviates from optimality by < 2.8 °C

Historical trends

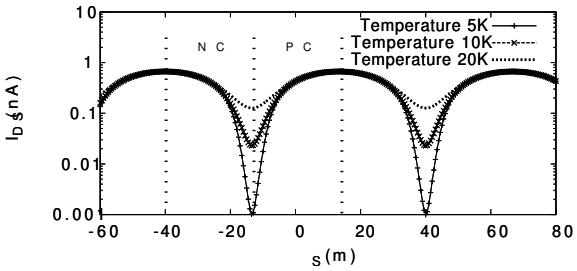


One possible future CMOS replacement



Projected switching energy: 1×10^{-18} J

SET I–V curve



Evaluated for embedded and high-performance use

Circuit design and modeling

Architectural reliability enhancements

Synthesized numerous processors in proposed architecture

ARM7, ASPIDA DLX, Jam RISC, LEON2 SPARC, Microblaze RISC, miniMIPS, MIPS, PLASMA, UCore, YACC, AES, AVR, CORDIC, ECC, FPU, RS, USB, and VC

Two orders of magnitude improvement in energy efficiency over CMOS

Summary of research philosophy

Attack problems at their points of greatest leverage, e.g.,

- Operating system for memory compression
- Circuits for low-power event-driven applications
- Design process for temperature-aware IC design

Modeling and design research are mutually-reinforcing

Some projects should ship soon, some should be risky

Acknowledgments

Advisees, for doing all the heavy lifting

Lan Bai, Zhenyu Gu, Sasha Jevtic, Ai-Hsing Liu, Lei Yang, Yonghong Yang, and Changyun Zhu

Collaborators

Srimat Chakradhar, Peter Dinda, Xiaobo Sharon Hu, Russ Joseph, Mat Kotowsky, Haris Lekatsas, Gokhan Memik, Li Shang, Huazhong Yang, and Hai Zhou

Acknowledgments

Mentors, for blunt and good advice since my arrival

Prof. Lawrence Henschen (my faculty mentor), Prof. Alok Choudhary (CES Division Head), and Prof. Abraham Haddad

Sponsors, for making our research possible

Dr. Helen Gill (NSF), Dr. Srimat Chakradhar (NEC Labs America), Dr. David Yeh (SRC), Dr. Anita LaSalle (NSF), and Dr. William Joyner (SRC)

Thank you for attending!

More information at http://www.eecs.northwestern.edu/~dickrp

Other upcoming Meet the EECS Faculty Seminars

- 4 May: Fabián Bustamante
  Grand Challenges in Large-Scale Distributed Systems
- 18 May: Dongning Guo
  Information and Estimation in Communications

Backup slides

Past work: Code compression

Lekatsas 2000, Xu 2004

- Store code compressed, decompress during execution
- Compress off-line, decompress on-line
- For RAM, less important than on-line data compression

Past work: Compression for swap performance

- Compressed caching
  - Douglis 1993, Russinovich 1996, Wilson 1999, Kjelso 1999
  - Add compressed software cache to VM
- Swap compression
  - RamDoubler, Cortez 2000, Roy 2001, Chihaia 2005
  - Compress swapped-out pages and store them in software cache
- Both techniques
  - Target: general purpose system with disks
  - Goal: improve system performance
  - Interface to backing store (disk)

Related work

Thermal modeling

- P. Li, L. T. Pileggi, M. Ashghi, and R. Chandra. Efficient full-chip thermal modeling and analysis. In *Proc. Int. Conf. Computer-Aided Design*, pages 319–326, November 2004
- Kevin Skadron, Mircea R. Stan, Wei Huang, Sivakumar Velusamy, Karthik Sankaranarayanan, and David Tarjan. Temperature-aware microarchitecture. In *Proc. Int. Symp. Computer Architecture*, pages 2–13, June 2003
- COMSOL Multiphysics (FEMLAB)

Thermal modeling

Lorenzo Codecasa, Dario D'Amore, and Paolo Maffezzoni. An Arnoldi based thermal network reduction method for electro-thermal analysis. *Trans. Components and Packaging Technologies*, 26(1):168–192, March 2003

Thermal-aware synthesis

- Rajarshi Mukherjee, Seda Ogrenci Memik, and Gokhan Memik. Temperature-aware resource allocation and binding in high-level synthesis. In *Proc. Design Automation Conf.*, June 2005
- W.-L. Hung, G. Link, Y. Xie, N. Vijaykrishnan, N. Dhanwada, and J. Conner. Temperature-aware voltage islands architecting in system-on-chip design. In *Proc. Int. Conf. Computer Design*, October 2005

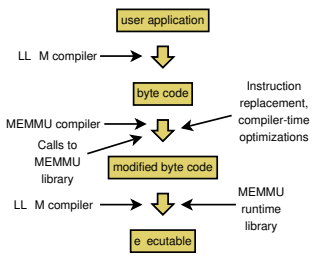
Wave propagation and update order

- Bound neighbor difference to prevent wave propagation problem

$$h_i' = \min\left(s_i(t_i), \min_{n \in N_i}(w \cdot (t_n + h_n - t_i))\right)$$

- w a small constant, e.g., 3
- Asynchronous times, which element to update?
- Discrete event simulator
- Used event queue ordered by earliest step target time $t_i + h_i$

Memory expansion for MMU-less embedded systems

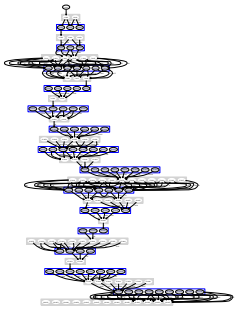


Observations and Results

- Main application: Sensor network nodes
- Implemented in LLVM and tested on TelosB nodes
- Increases usable memory by 50%, no changes to applications
- Performance and energy penalties small after compiler optimizations
- CASES'06

With Lan Bai and Lei Yang

Application characterization for system synthesis



Applications

- Extract communication graphs from arbitrary multithreaded applications
- Non-intrusive
- Use for application-specific multiprocessor synthesis
- CODES-ISSS'06
- Publicly released

With Ai-Hsin Liu