Software Architectural Transformation: A New Approach to Low Energy Embedded Software

T.K. Tan, A. Raghunathan, and N.K. Jha

Design, Automation, and Test in Europe(DATE) 2003

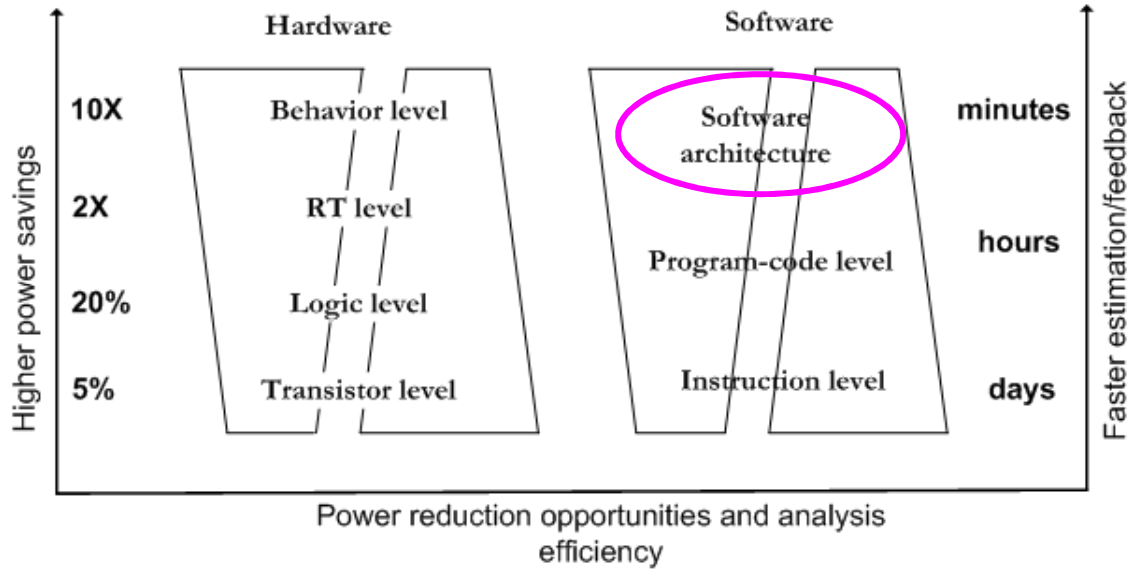
2008. 5. 7
AhRim Han

KAIST SE LAB
KAIST Software Engineering Laboratory

Contents

Introduction(1/2)

- Low power techniques at various levels of design abstraction



- **High abstraction level**
  → Power saving is more efficient and power estimation is faster

- **Software domain**
  → Previous studies only focused in instruction-level and compiler optimizations

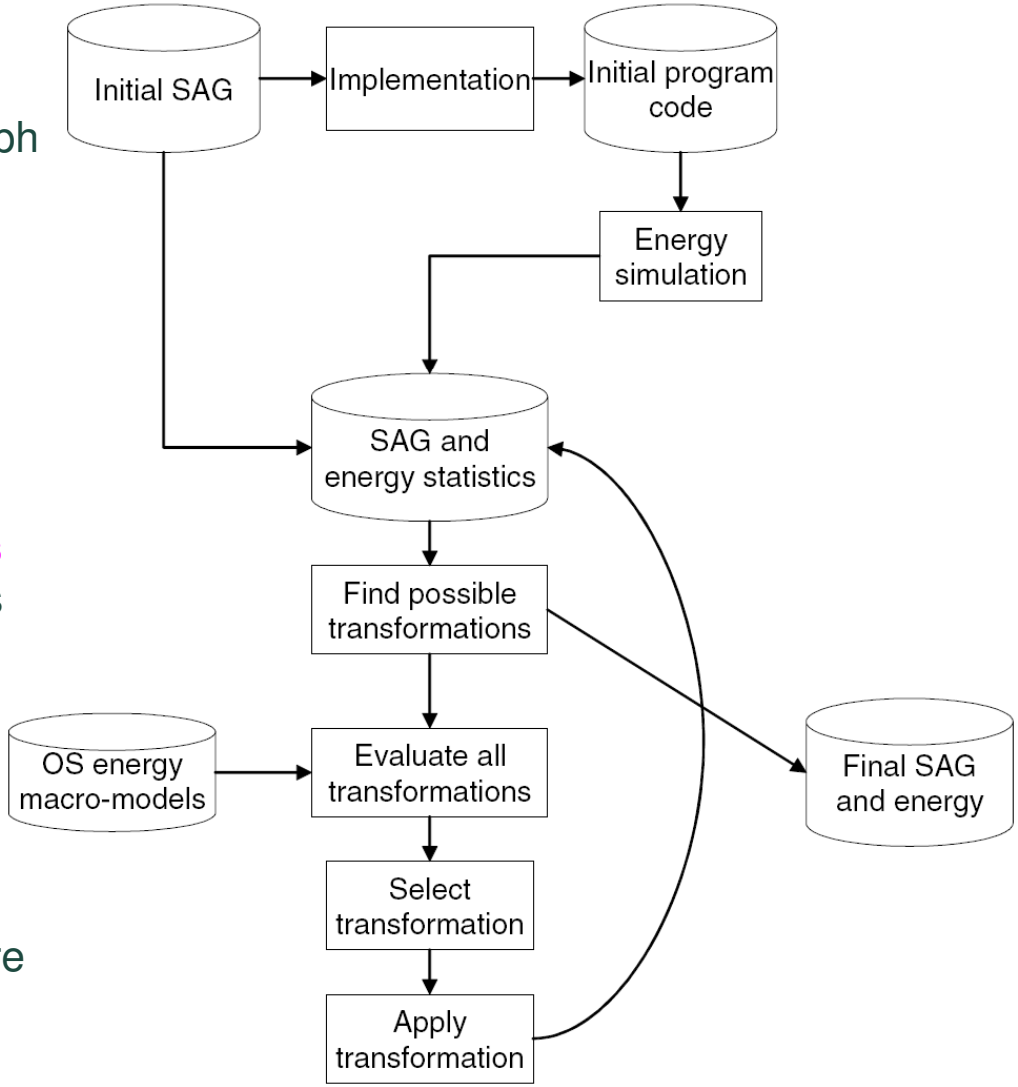
Introduction(2/2)

- Factors that affect power consumption in a software architecture level
  - Software architectural style selection
  - Software architecture design: components & connectors
- In this paper,
  - Provide a systematic methodology for applying software architectural transformations to derive an power-optimized architecture for given embedded software
    - Architecture style - OS-driven multi-process architectural style
    - Components - application processes, signal handlers, device drivers, etc.
    - Connectors - inter-process communication (IPC), synchronization mechanisms

Overview of the approach

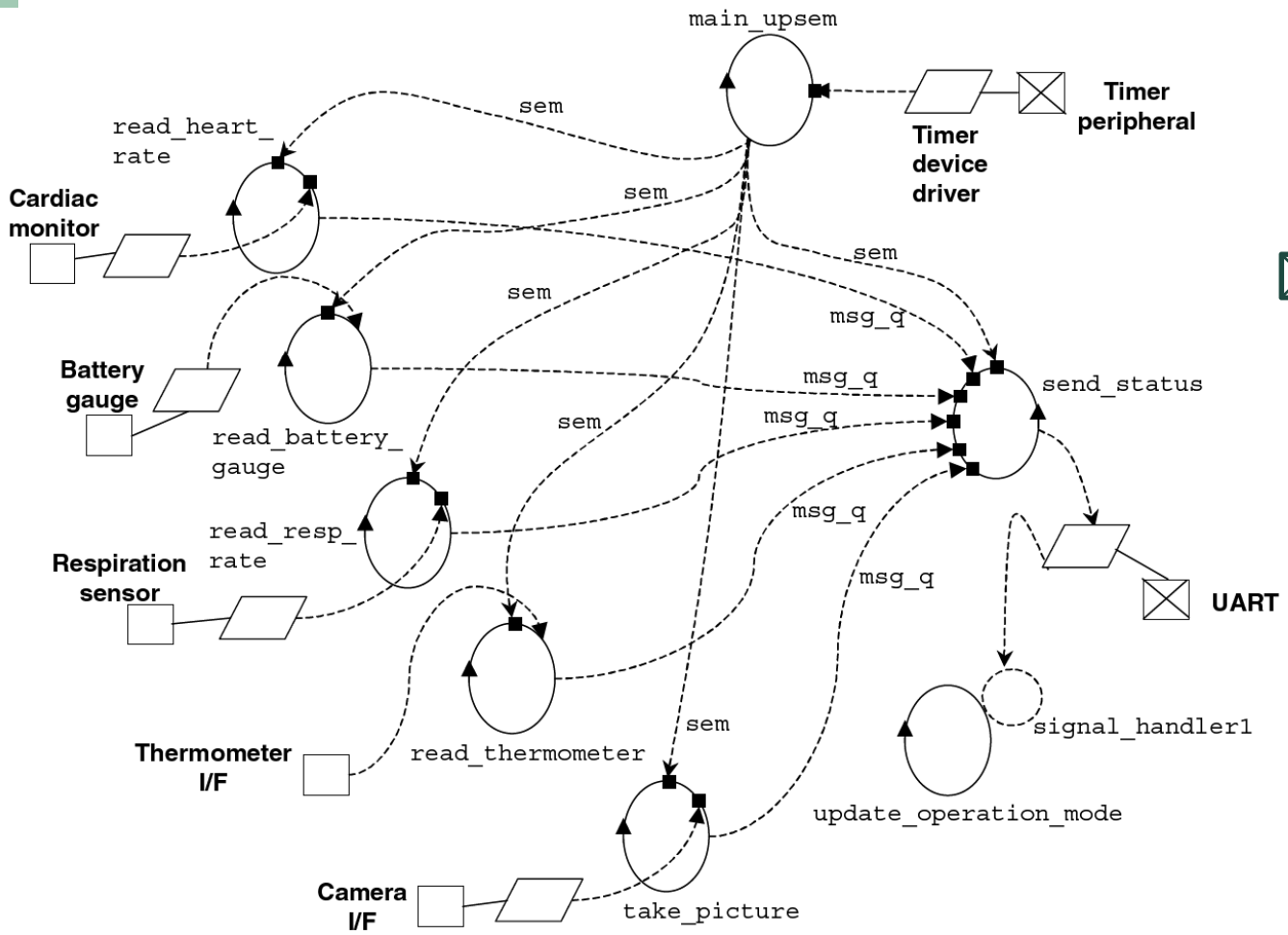
- Software architectural level energy minimization methodology

1. Constructing a software architecture graph (SAG) to represent a program
2. Deriving an initial profile of the energy consumption and execution statistics using a energy simulation framework
3. Evaluating the energy impact of atomic **software architecture transformations** through the use of energy macro-models
4. Constructing sequences of atomic transformations that result in maximal energy reduction
5. Generation of a program source code to reflect the optimized software architecture



Software architecture graph (SAG)

- SAG for a situational awareness system



Hardware device

Active hardware device

Device drivers

Association

Application processes

Signal handlers

IPC mechanism

Communication of data or control messages

Blocking communication - if none, Non-blocking

OS energy macro-models(1/4)

- Definition
  - A function expressing the relationship between
    - Energy consumption of a system function and predefined parameters
  - Example
    - Energy macro-models in Linux

Sub-systems	System Functions	Macro-models (nJ)	Remarks
Process Manager	fork()	$E_{parent} = 30540$	$\epsilon = 3.6\%$
Process Manager	fork()	$E_{child} = 4300$	$\epsilon = 0.1\%$
Process Manager	getpid()	$E = 177$	$\epsilon = 0.1\%$
IPC	msgsnd(x bytes)	$E = 4752 + 1.08x$	$\epsilon = 1.4\%$
IPC	msgrcv(x bytes)	$E = 4913 + 1.19x$	$\epsilon = 1.3\%$
IPC	pipe write(x bytes)	$E = 1147 + 1.64x$	$\epsilon = 11.2\%$
IPC	pipe read(x bytes)	$E = 690 + 1.65x$	$\epsilon = 14.4\%$
IPC	signal()	$E = 2739.9$	$\epsilon = 0.1\%$
Timer Functions	gettimeofday()	$E = 495$	$\epsilon = 0.1\%$
Timer Functions	settimeofday()	$E = 595$	$\epsilon = 0.1\%$
Memory Manager	calloc(x blocks, size y)	$E = 287 + 0.986xy$	$\epsilon = 5.1\%$
Memory Manager	malloc()	$E = 123$	$\epsilon = 2.0\%$
Memory Manager	free()	$E = 160$	$\epsilon = 27.5\%$
Memory Manager	memcpy(x bytes)	$E = 110 + 0.98x$	$\epsilon = 4.6\%$
File System	open()	$E = 2351$	$\epsilon = 2.1\%$
File System	close()	$E = 696$	$\epsilon = 0.1\%$
File System	file read(x bytes)	$E = 1209 + 1.06x$	$\epsilon = 16.9\%$
File System	file write(x bytes)	$E = 2430 + 2.75x$	$\epsilon = 15.7\%$

- Show 5.9% average error with respect to the energy data used to obtain the macro-models

OS energy macro-models(2/4)

- How to make
  - Use a regression technique
    - Ex) Energy of a system function ***msgsnd***()
      - $E = c_1 + c_2 x$ (x – *size of bytes for message sending*)

$$\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ .. & .. \\ 1 & x_n \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ .. \\ E_n \end{pmatrix}$$

OS energy macro-models(3/4)

- Where to be used
  - Evaluating energy impact of software architectural transformations
    - Configuration (C) of a software architecture is transformed
      - $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow \ldots \rightarrow C_{Optimal}$
    - It takes too much time to obtain an energy of each C by a detailed system simulation framework
      - $E(C_0)$, $E(C_1)$, $E(C_2)$, … , $E(C_{optimal})$
    - The effect of software architectural transformations is to alter the manner in which OS services are employed
    - *Energy change estimates* can be obtained by OS macro-models
    - Energy of a transformed software architecture can be evaluated
      - $E(C_0)$ by performing a detailed energy profiling one time
      - $E(C_1) = E(C_0) + \triangle E(\mathrm{T}_{\mathrm{C0}\rightarrow\mathrm{C1}})$, $E(C_2) = E(C_1) + \triangle E(\mathrm{T}_{\mathrm{C1}\rightarrow\mathrm{C2}})$, … , $E(C_{optimal}) = E(C_{optimal\_pre}) + \triangle E(\mathrm{T}_{Coptimal\_pre\rightarrow Coptimal})$

OS energy macro-models(4/4)

- Where to be used (Cont'd)
  - Calculating energy change estimates
    - Ex) Amount of energy change incurred by replacing IPC_1 with IPC_2

• Energy macro models for IPC_1 and IPC_2

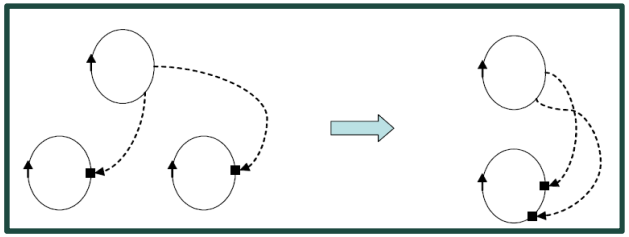
$$E_{ipc1}(x) = c_{11} + c_{12}x$$
$$E_{ipc2}(x) = c_{21} + c_{22}x$$

• Amount of energy change estimates

$$\triangle E$$
$$= [E_{ipc2}(x) - E_{ipc1}(x)]\, N_{ipc}$$
$$= [(c_{21} - c_{11}) + (c_{22} - c_{12})\, x]\, N_{ipc}$$

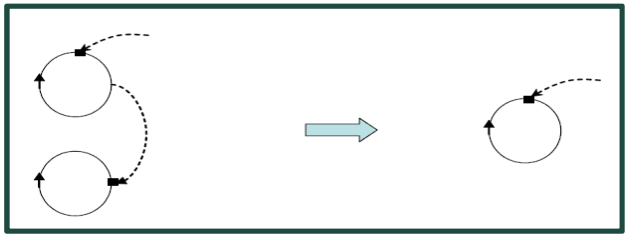
Software architectural transformation(1/3)

- Temporal cohesion driven process merging



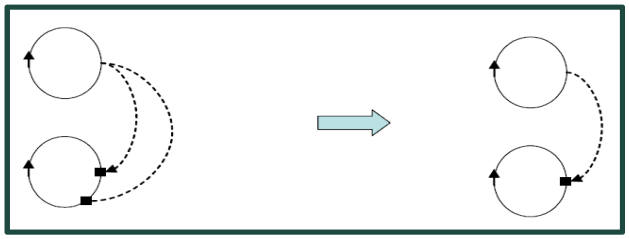
$$\Delta E = -E_{ctx} N_{ctx}$$

- Sequential cohesion driven process merging



$$\Delta E = -E_{ipc}(x) N_{ipc} - E_{ctx} N_{ctx}$$

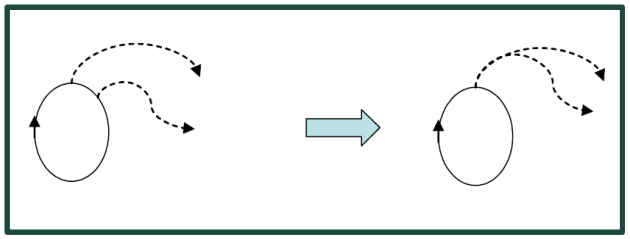
- Temporal cohesion driven IPC merging



$$\Delta E = -E_{ipc\_read}(x) N_{ipc\_read}$$

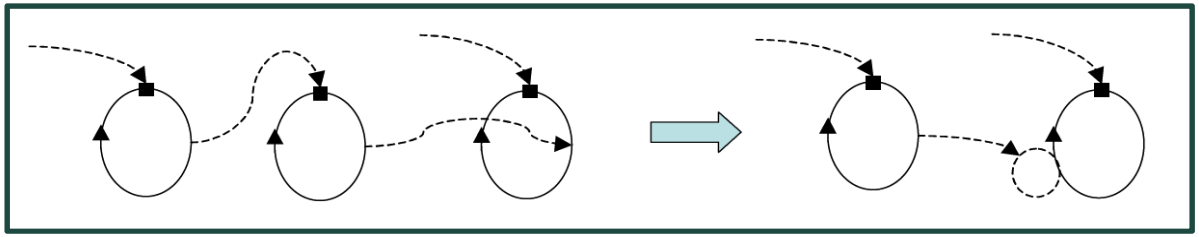
Software architectural transformation(2/3)

- Intra-process computation migration



$$\Delta E = -\left(E_{ipc\_wr}(x) + E_{ipc\_wr}(y) - E_{ipc\_wr}(x+y)\right) N_{ipc\_wr}$$

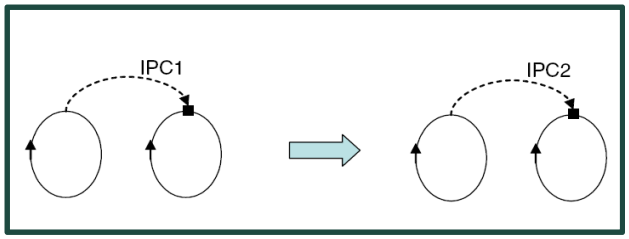
- Process embedding as signal handlers



$$\Delta E = -E_{ctx} N_{ctx} + E_{sig} N_{sig}$$

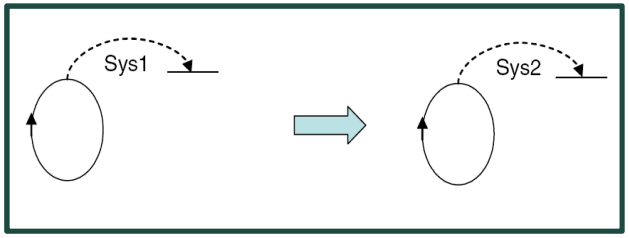
Software architectural transformation(3/3)

- IPC replacement



$$\Delta E = [E_{ipc2}(x) - E_{ipc1}(x)]\, N_{ipc}$$

- System function replacement



$$\Delta E = [E_{sys2} - E_{sys1}]\, N_{sys}$$

Case study(1/3)

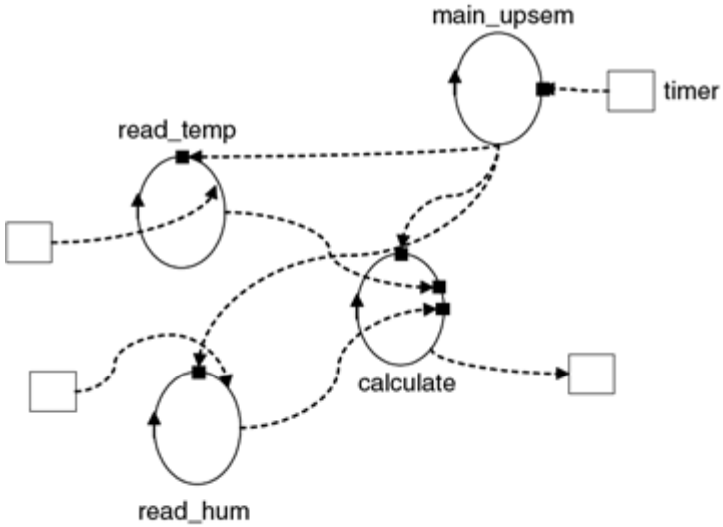
- Experiment environment
  - Embedded Linux OS: arm-linux v2.2.2
  - Hardware platform: StrongARM embedded processor
  - Energy simulation framework: EMSIM [1]
  - High-level energy macro-models of the Linux OS: [2]
- Studied various multi-process example programs

Program name	Description
Aware	Situational awareness system illustrated in the previous example
Headphone	Program used in an audio headset application
Vcam	Embedded software from a video camera recorder
Climate	Application for collecting and processing climate information
Navigator	Software from a global navigation
ATR	Part of an automatic target recognition program

[1] T. K. Tan et al., "A simulation framework for energy-consumption analysis of OS-driven embedded application," IEEE Trans. on CAD of Integrated Circuits and Systems(TCAD), 2003
[2] T. K. Tan et al., "High-level energy macromodeling of embedded software," IEEE Trans. on CAD of Integrated Circuits and Systems(TCAD), 2002

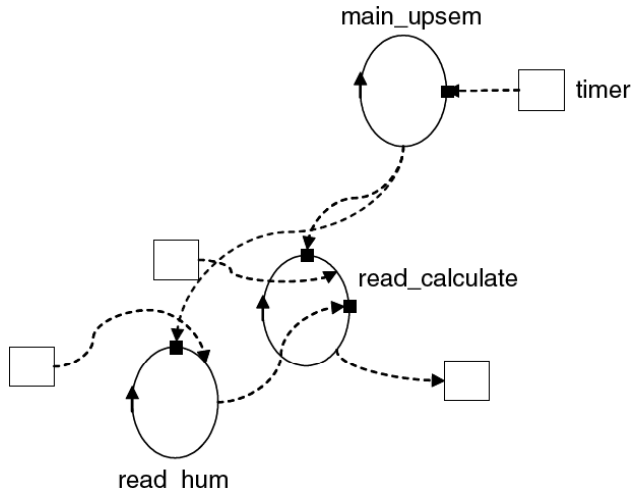
Case study(2/3)

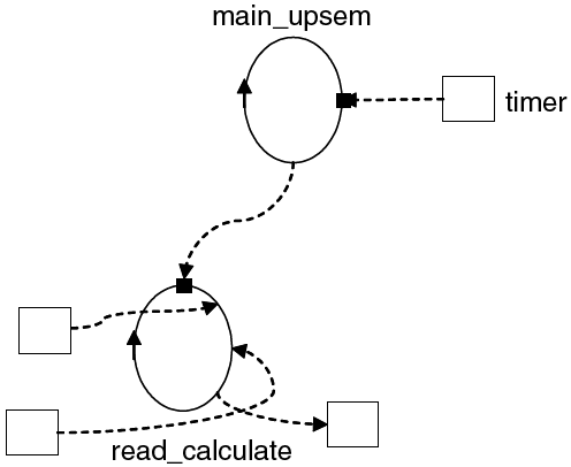
- Energy minimization: ex) climate system



Initial SAG

$E_0 = 0.239mJ$





Final SAG

$E_0 = 0.081mJ$

∴ % Energy reduction = 66.1%

Case study(3/3)

- Results of energy minimization

Examples	Original			Optimized			% Energy reduction
	Energy (mJ)	# proc	# ipc	Energy (mJ)	# proc	# ipc	
Aware	12.956	8	11	8.204	7	9	36.7%
Headphone	1.668	6	8	1.461	3	2	12.4%
Vcam	1.572	4	5	1.375	2	1	12.5%
Climate	0.239	4	5	0.081	2	1	66.1%
Navigator	1.659	5	7	1.456	3	3	12.2%
ATR	6.940	4	7	6.199	3	4	10.7%

Software architecture level techniques
for energy reduction can significantly extend and complement
existing low power software design techniques!!!

Related work

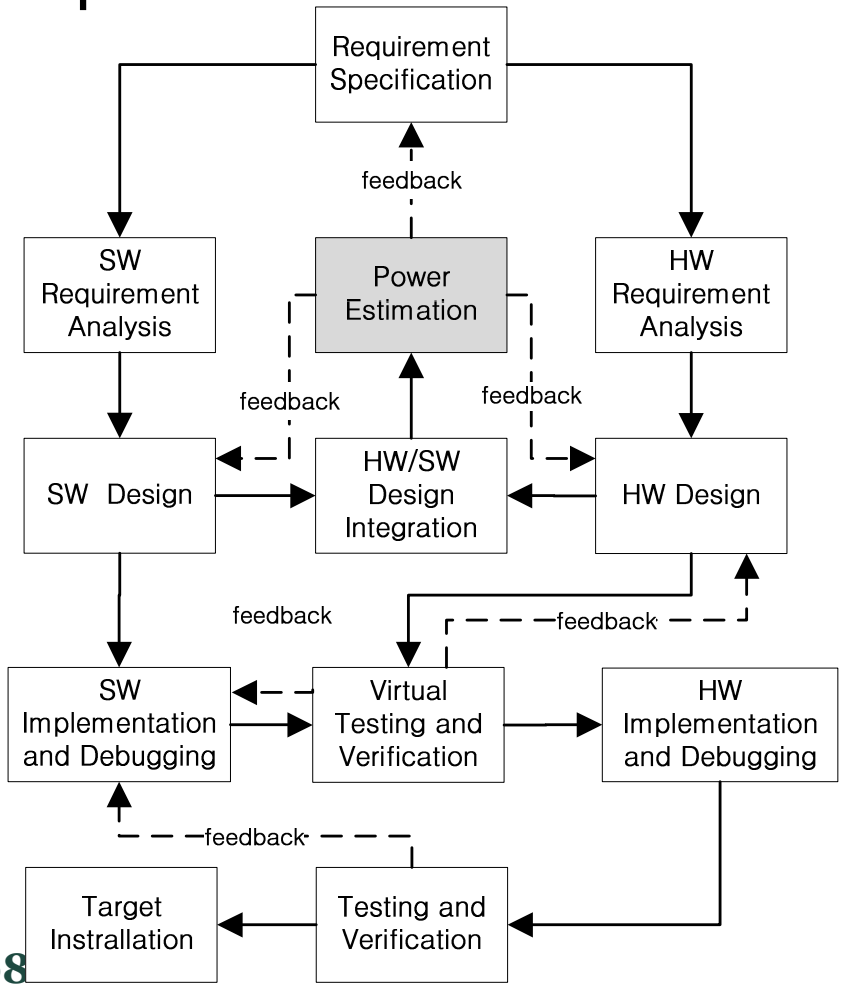
- Energy minimization techniques at various level
  - Software design
    - Instruction
      - Instruction selection, register allocation, instruction scheduling (e.g., loop optimization by merging and unrolling), etc.
    - Program code
      - Hot operations (e.g., external memory R/W, some coprocessor operations) reduction, data representation change, etc.
  - Hardware design
    - Transistor/Logic
      - Capacitance minimization/low leakage, low voltage, etc.
    - Architecture/System
      - Resource scheduling (e.g., parallel processing, pipelining, shut down)

Conclusion

- Provide a method to design a low power embedded software by software architectural transformations
  - Constructing a software architectural graph (SAG)
  - Deriving initial energy statistics using a detailed energy simulation framework
  - Constructing sequences of atomic software architectural transformations guided by energy change estimates derived from OS energy macro models

Model-based power estimation(1/4)

- Power estimation point in the embedded system development process



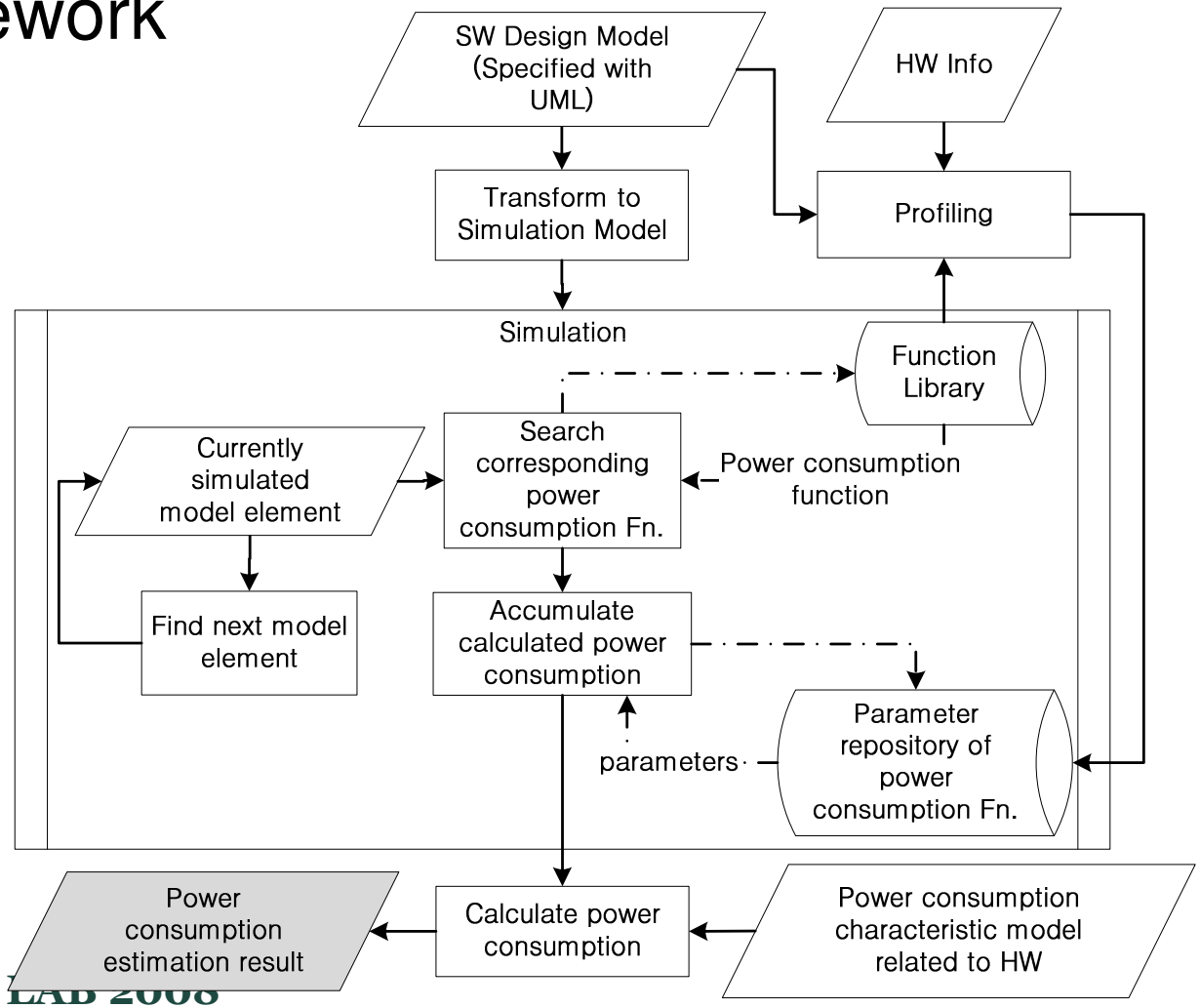
Model-based power estimation(2/4)

- Energy estimation studies at various level

Type	Level	Author	Method	Tool
HW	Transistor	Najm, etc.	• Sum of the Power {Dynamic (Transistor switching), Short circuit, Leakage current, …}	SPICE PowerMill
	Logic			
	Gate			
	Circuit			
	RT	Landman, Brooks, Bogliolo, etc.	• Sum of the Power {Adders, Multipliers, Controllers, Register files, SRAM's, …}	Simple-Power Wattch
	Architecture			
	Behavior	Kumar, Raghunathan, etc.	• Sum of the Power {Processor, Bus, Memory, …}	PowerPlay
	System	Lidsky, etc.		
SW	Instruction	Tiwari, etc.	• Sum of the Power {Instruction base cost, Inter-instruction cost, Other factors' (Cache miss, Pipeline stall, etc.) cost, …}	JouleTrack Panalyzer
	Function	Qu, etc.	• Sum of the Power {Built-in library functions, User-defined functions, Main functions, …}	-

Model-based power estimation(3/4)

- Model-based power estimation simulation framework



Model-based power estimation(4/4)

- Applications of the power estimation technique
  - Develop low power design patterns
  - Identify and visualize the bottleneck of power consumption in an embedded software design model

Thank You .