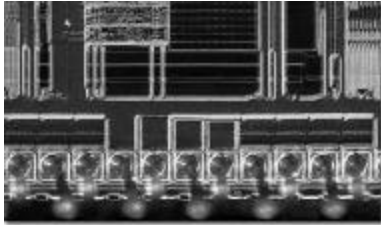


# Power Analysis of Embedded Software

by Vivek Tiwari, Sharad Malik, and Andrew Wolfe



February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

1

## Introduction...

- The use of embedded systems is constantly increasing.
- Part of this increase is due to the switch from application specific logic to application specific code running on existing processors.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

2

## Introduction...

- This change is driven by two distinct forces:
  - The increasing cost of setting up and maintaining a fabrication line
  - Increased pressures to reduce the time to market and maintain a reliable schedule
- This is why we are seeing the basic unit of computation change from the logic gate to an instruction running on an embedded processor.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

3

## Introduction...

- Power constraints form an important part of the design specification for most embedded systems. aka: power critical.
- This has led many researchers to focus on power estimation and low-power designs.
- Unfortunately, there is little available to help embedded system designers evaluate their designs in terms of the power metric.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

4

## Introduction...

- The embedded processors currently used in designs usually take one of two routes:
  - “off the shelf” processors or DSPs
  - Embedded cores which can be incorporated into a larger chip with other logic and memory

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

5

## Introduction...

- For DSPs, the designer only has the processor information the manufacturer makes available through the data books.
- For the embedded cores, the designer only has logic/timing simulation models to help verify the designs.
- In both cases there is no lower level information available for power analysis.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

6

## The Purpose

- Power consumption has been a subject of intense study, but the previous research has adopted a “bottom-up” approach, using detailed layouts and sophisticated power analysis tools, which are expensive and time consuming.
- However, no attempts have been made to relate the power consumption to the software that executes it.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

7

## Purpose

- It is recognized that that the power consumption of the processor varies from program to program, but there is a lack in tools to analyze this variation.
- The purpose of the research presented in this paper is to overcome these deficiencies by developing a methodology to easily analyze the power consumption from the execution of a given program.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

8

## How the Purpose was met and the logic behind it

- The purpose was met by measuring the current drawn by the processor as it repeatedly executes certain instructions or short instruction sequences.
- The reason this works is due to that fact that despite modern processors being extremely complex systems of several interacting blocks the internal complexity is hidden behind a simple interface—the instruction set.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

9

## How the Purpose was met and the logic behind it

- To model energy consumption of a complex system we calculate the power cost of each instruction. Also, in any given program there are also inter-instruction effects, such as the the effect of the circuit state, pipeline stalls, and cache misses.
- Thus the sum of the power costs of each instruction plus the power cost of the inter-instruction effects can be an accurate estimate for the power cost of a given program.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

10

## Quick Math Summary

- The average power is equal to the product of the average current and the supply voltage.  
 $P = I * V_{cc}$ .
- Since power is the rate at which energy is consumed, the amount of energy consumed is equal to the product of Power and execution time.  
 $E = P * T$ .
- Finally, the execution time is equal to the product of the number of clock cycles and the clock period.  $T = N * t$ .

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

11

## How the current was measured

- For this study a 40 MHz Intel 486DX2-S CPU was used with 4MB of DRAM.
- Although the numbers in this report are specific to this processor, the methodology used in the model is widely applicable.
- The current was measured using a standard, dual-slope integrating digital ammeter.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

12

## How the current was measured

- The programs being considered were put in infinite loops and current readings were taken.
- The main limitation of this approach is that it will not work for programs with larger execution times since the ammeter may not show a stable reading. Since the main use of this approach was in determining the current drawn during a particular instruction, this isn't much of a problem.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

13

## Base Energy Cost

- The base cost for an instruction is determined by constructing a loop with several instances of the same instruction. The average current being drawn is then measured. This current is then multiplied by the number of cycles taken by each instance of the instruction.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

14

## Base Cost Examples

- Here is an example of CPU base costs for some of the instructions. The numbers in Column 3 are the observed average current values. The overall base energy cost of an instruction is the product of Column 3, 4, and the constants  $V_{cc}$  and  $T$ .

TABLE I  
SUMMARY OF THE BASE COST VALUES FOR THE 486/487

Number	Instruction	Current (mA)	Cycles
1	ADD	215.7	3
2	MOV 32, 32	215.7	3
3	MOV 32, (32)	416.4	3
4	MOV 32, (32), 32	416.4	3
5	MOV 32, 32, 32	215.7	3
6	MOV 32, (32), 32	416.4	3
7	ADD 32, 32	113.0	3
8	ADD 32, (32)	420.0	3
9	ADD 32, 32, 32	113.0	3
10	SHL 32, 1	385.0	3
11	SHL 32, 32	385.0	3
12	SHR 32, 1	385.0	3
13	SHR 32, (32)	385.0	3
14	AND 32, 32	173.0	3
15	OR 32, 32	173.0	3
16	XOR 32, 32	173.0	3
17	AND 32, 32, 32	173.0	3
18	OR 32, 32, 32	173.0	3

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

15

## Base Energy Cost

- It is important not to oversize the loops that are used to determine the base costs of your program.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

16

## Inter-Instruction Effects

- When sequences of instructions are considered, certain inter-instruction effects come into play, which are not reflected in the cost computed solely from base costs.
- Here are the three areas in which this occurs; circuit state, resource constraints, and cache misses. This is an overview since the paper becomes slightly more involved.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

17

## Inter-Instruction Effects: circuit state

- The switching activity in a circuit is a function of the present inputs and the previous state of the circuit. Thus, it can be expected that the actual energy cost of executing an instruction in a program may be different from the instruction's base cost. This is because the previous instruction in the given program and in the program used for base cost may be different.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

18

## Inter-Instruction Effects: circuit state

- For Example, consider this loop:  
XOR BX, 1  
ADD AX, DX
- The base costs of the XOR and ADD instructions are 319.2 and 313.6 mA. The expected base cost would be their average, 316.4, but in actuality the cost is 323.3. This is because the base costs are determined while executing the same instruction over and over again.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

19

## Inter-Instruction Effects: circuit state

- The cost of a pair of instructions is always greater than the base cost of the pair and the difference is termed the circuit state overhead.
- On a final note, after extensive study it was found the circuit state overhead has a limited range—between 5.0 mA and 30.0 mA and most frequently is around 15.0 mA.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

20

## Inter-Instruction Effects: Resource Constraints

- Resource constraints in the CPU can lead to stalls e.g. pipeline stalls and write buffer stalls.
- These can be considered as another kind of inter-instruction effect since they cause an increase in the number of cycles needed to execute a sequence of instructions.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

21

## Inter-Instruction Effects: Resource Constraints

- The energy cost of each kind of stall is determined through experiments that isolate the particular kind of stall.
- For example, an average cost of 250 mA for stall cycles was determined for the prefetch buffer stall.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

22

## Inter-Instruction Effects: Resource Constraints

- It has been observed that the cost of stalls can show some variation depending upon the instructions involved in the stall.
- However, in general the use of a single average cost value for each stall type is sufficient.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

23

## Inter-Instruction Effects: Resource Constraints

- To account for the energy cost of the stalls during program cost estimation, the number of stall cycles has to be multiplied by the experimentally determined stall energy cost. This product is then added to the base cost of the program. The number of stall cycles is estimated through a traversal of the program code.

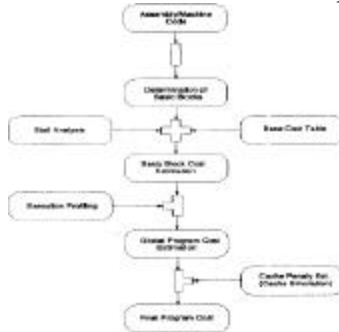
February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

24



## Final Overview of Technique



February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

11

## Optimization Note

- While the reordering of a given set of instructions in a piece of code may have a limited impact on the energy cost, the choice of which instructions are used in the generated code can significantly affect the cost.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

12

## Summary

- This paper presents a methodology for analyzing the energy consumption of embedded software.
- The motivation for the analysis is three-fold.
  - It provides insight into the energy consumption in processors.
  - It can be used to help verify if an embedded design meets its energy constraints and guide the development so that it does meet the constraints.
  - Attempts at code re-writing demonstrate significant power reductions—justifying the motivation for such a power analysis technique.

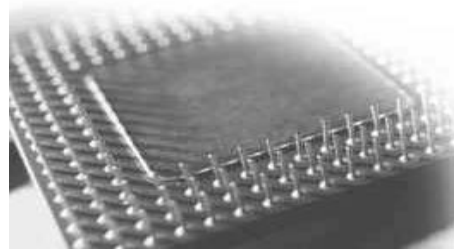
February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

13

## System and architecture-level power reduction of microprocessor-based communication and multi-media applications

By Lode Nachtergaele, Vivek Tiwari, Nikil Dutt



February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

14

## Introduction...

- Current microprocessor architectures have become dominated by the data access bottleneck in the cache, system bus and main memory subsystems. These systems also have a large influence on the systems power consumption.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

15

## Introduction...

- In order to provide high data throughput at reasonable power consumption for these demanding applications, novel solutions for the memory access and data transfer will have to be introduced. These will have to be both at the processor architecture and the compiler level.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

16

## Introduction...

- The question this paper addresses in this paper is what would these solutions look like.
- The paper shows that these solutions will be based on processor architecture optimizations, sophisticated application of compiler technology, and exploiting the interface between system hardware/software.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

37

## Architecture Optimizations

- Due to the dependence of power on voltage, voltage reduction is the most favored method of reducing power.
- It has been shown that aggressive voltage reductions are possible if architectural and algorithmic transformations are applied to the problem (pipelining and parallelism) to regain the lost performance of voltage reduction.
- This works well for throughput-oriented limited-function applications(e.g. digital filtering).

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

38

## Architecture Optimizations

- More recently, architectural optimizations aimed primarily at power reduction have become an active area of research.
- Here are the main ideas classified by theme.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

39

## Module Parameter Tradeoffs

- This configures the caches, register-files, etc. to the optimal size for the desired power/performance.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

40

## Exploiting locality both for instructions and data

- Creating mini-caches and mini-TLBs to avoid the cost of looking up the larger main cache.
- Value locality which saves the most recent computations to avoid re-computation.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

41

## Enabling more power down

- Partitioning the cache to allow one necessary bank to be powered up and word-width wise partitioning of data paths.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

42

## Speculation Reduction

- Dynamically reducing the speculation in the machine to reduce power—e.g. limiting instruction issue if the number of predicted branches exceeds a limit.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

43

## Hardware hooks to allow for more software control on power

- A loop cache into which basic blocks are statically allocated by the compiler.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

44

## Architecture Optimizations

- Optimizations such as the last slides are local to the CPU. Power reduction techniques of a wider scope are possible if the CPU is seen as a component of an overall system.
- This allows for each component to be powered-up or down whenever appropriate.
- This motivates the application of dynamic power management systems.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

45

## Architecture Optimizations

- For embedded applications, there are opportunities for additional flexibility and power management systems tuned for specific applications be extremely efficient means for power reduction.
- Improved modeling of system behavior has gained a lot of attention lately, along with improved power management policies.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

46

## Architecture Optimizations

- An additional source of power efficiency comes from extending power management to include control on the CPU's voltage and performance.
- Dynamic voltage/freq has high potential, but required a unified hw/sw approach.
- Multi-media applications are ideally suited for dynamic voltage/freq scaling since they often have regular activity patterns that can be pre-characterized.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

47

## Optimized Platform Mappings

- In the domain of algorithm transformations and compilation technology for embedded data-dominated applications, there has been a lot of work for the traditional metrics of cost and performance.
- Decision made at this stage heavily influence the final outcome when the appropriate architectural issues of the embedded memories are correctly incorporated.

February 06, 2002

UCR CS209: Hardware/Software Engineering of Embedded Systems

48

## Optimized Platform Mappings

- This has to happen at the instruction-level parallelism compiler and in the preceding system compilation stages.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

49

## System-level Code Transformations

- Exploration of Data Transfer and Storage (DTS) is an important pre-compilation step.
- The reduction of size and number of transfers decrease both the power consumption of the memory system while preserving the behavior.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

50

## System-level Code Transformations

- The major principles of source-to-source transformations of the DTSE methodology are:
  - Global data-flow transformation to avoid redundant transfers.
  - Global loop and control flow transformations to increase locality of reference.
  - Data reuse exploration to exploit the available memory hierarchy.
  - SDRAM memory organization.
  - Data layout decisions to reduce the memory size and improve the cache hit rates.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

51

## Platform Compiler Technology

- Early experiments demonstrated reduced energy consumption through improved register allocation, resulting in fewer spills to memory.
- Compiler techniques that improves data locality through coarse-grain transformations and data layout optimization, result in significantly fewer cache misses, leading to improved performance and lower power dissipation.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

52

## Platform Compiler Technology

- Similarly, instruction scheduling techniques to reduce instruction cache misses have been developed, resulting in reduced bus transition per off-chip memory transfer.
- Recent work in memory-aware compilation aims to better exploit memory access protocols of contemporary DRAMs for improving the memory bandwidth of applications.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

53

## Platform Compiler Technology

- The effects of such compiler optimizations on power dissipation require a comprehensive measurement or simulation environment, since the relationship between performance and power or energy is not easily predictable.

February 06, 2002

UCR CS209: Hardware-Software Engineering of Embedded Systems

54

## Platform Compiler Technology

- Finally, compiler-controlled power management techniques are beginning to appear, that dynamically tradeoff power for performance. The compiler, through a combination of static analysis, profile-driven data and feedback driven optimization, can thus modify the power/performance characteristics of the target architecture, in consort with system-level power management schemes.