Cycle-Accurate Energy Measurement and Characterization With a Case Study of the ARM7TDMI

Naehyuck Chang, Kwanho Kim, and Hyung Gyu Lee

***Abstract*—Energy characterization is the basis for high-level energy reduction. Measurement-based characterization is accurate and independent of model availability and is thus suitable for commercial off-the-shelf (COTS) components, but conventional measurement equipment has serious limitations in this context. We introduce a new technique for the energy characterization of a microprocessor, using a cycle-accurate energy measurement system based on charge transfer which is robust to spiky noise and is able to collect a range of energy consumption profiles in real time. It measures the energy variation of the CPU core by changing the instruction-level energy-sensitive factors such as opcodes (operations), instruction fetch addresses, register numbers, register values, data fetch addresses and immediate operand values at each pipeline stage. Using the ARM7TDMI RISC processor as a case study, we observe that the energy contributions of most instruction-level energy-sensitive factors are orthogonal to the operations. We are able to characterize the energy variation, preserving all the effects of the energy-sensitive factors for various software methods of energy reduction. We also demonstrate applications of our measurement and characterization techniques.**

***Index Terms*—Energy characterization, energy measurement, low-power design, microprocessor.**

I. Introduction

Low energy consumption has emerged as a major performance metric for digital systems, motivating many low-level contributions toward cool chips and cool systems. Over the years, designers have also become interested in high-level and software-level energy reduction techniques—just as, with motor vehicles, better fuel consumption is achieved not only by developing efficient designs but also by formulating efficient driving methods. High-level energy-reduction studies often focus on optimization, assuming that energy characteristics are fixed. But energy characterization can itself be the basis of high-level energy-reduction techniques, because optimization policies do not rely totally on a specific physical design.

Microprocessors consume significantly different amounts of energy during each clock cycle. In a programming model, the energy variation is dependent on *instruction-level energy-sensitive factors* such as instruction fetch addresses, opcodes (operations), register encoding, data fetch addresses, immediate operands, and so on. Software energy reduction optimizes energy consumption by changing the energy-sensitive factors while preserving the semantics of the original design. Previous work on energy characterization has focused on average power analysis, which is useful to estimate total consumption, but is inadequate for high-level reduction techniques. In fact, it is not easy to achieve complete energy characterization using conventional approaches. This paper introduces a new measurement-based energy characterization for microprocessors to fulfill that requirement.

We present a real-time cycle-accurate energy measurement technique for digital systems. We characterize the energy variation of a COTS microprocessor at the instruction level with respect to the energy-sensitive factors. Unlike previous characterizations, ours does not average out the energy variation and it is therefore a useful basis for high-level energy reduction techniques, including opcode or register re-encoding, address relocation and instruction rescheduling.

We demonstrate the new method through a case study of the ARM7TDMI RISC core. This study was made possible by an in-house measurement tool with a real-time acquisition ability that can perform a large number of measurements over a short period.

The rest of this paper is organized as follows. A literature survey is presented and the motivation of this work are described in Section II. Section III introduces our real-time cycle-accurate energy measurement system, and Section IV presents the results for the ARM7TDMI core. In Section V we describe energy characterization for high-level power reduction, and Section VI suggests applications of the characterization. Section VII concludes the paper.

II. Related Work

We may acquire energy consumption profiles by simulation or measurement. Energy simulation is convenient provided that a simulation model is available, because it does not necessitate a prototype. Simulation is also preferable as energy consumption can vary with bus configuration and peripheral devices. Related studies [1]–[4] have described high-level processor simulators and estimated average power consumption at reasonable complexity. Low-level energy simulation is often used to back up high-level simulation [5]. Alternatively, a black-box model may be introduced to when simulation models for peripheral devices are not available [4]. Recently [6], a power simulator has been used to give a system-wide view, including a microprocessor and a memory system; however, for the most part, microprocessor-level simulators do not reflect real implementation of commercial off-the-shelf (COTS machines).

We need a working prototype to perform energy measurement. Even with a prototype, correct measurements are not easily obtainable because digital systems consume energy in a spiky manner, at frequencies of hundreds of MHz in the power spectrum [7]. Digital multimeters (DMMs) [8], [9] can only measure average power due to their limited bandwidth. The use of an oscilloscope overcomes this drawback [10], but the energy calculation procedure is invariably error-prone.

Measurement-based characterization would be promising if we could bypass time-consuming conventional techniques, which restrict the feasible number of experiments and thus make it difficult to construct a sufficiently detailed sample space for energy characterization. Admittedly, measurement-based characterization will always be system-dependent, but we certainly need system-specific behavior for real reduction practices, and avoiding the need for a model is attractive.

In spite of the limitations of existing energy estimation methods, some work has been done on power characterization for high-level power reduction. Intensive measurement-based characterizations [8], [9], [5] have been used to determine an *instruction base cost* and an *inter-instruction cost*, and to demonstrate energy reduction in a DSP application [9]. Conventional equipment requires patient experiment even to achieve this level of characterization, and the inter-instruction costs average out the energy consumption variation due to different energy-sensitive factors. So this scheme does not afford many alternative plans for reduction, although it is useful for average power estimation.

Another intensive simulation study [5] introduces a limited analysis of average power variation due to addressing modes and data bus activities; and an operand-dependent power analysis has also been introduced [1], which takes into account the power costs of representative components. This work excludes many significant components and the results associate different costs with components when processing different instructions, results which are inevitably difficult to confirm.

Manuscript received January 12, 2001; revised October 8, 2001. This work was supported in part by the Brain Korea 21 Project under SNU RIACT research project contract. An earlier version of this paper was presented at ISLPED 2000.

The authors are with the School of Computer Science and Engineering, Seoul National University, Korea (e-mail: naehyuck@snu.ac.kr).

Publisher Item Identifier S 1063-8210(02)00469-9.

Other work creates different levels of abstraction, which may be useful for hardware designers [11], [12], and for the implementors of higher-level software such as power management [13], [14].

Furthermore, existing energy characterizations assume that microprocessors are designed in static CMOS . Consequently, it is also assumed that all the energy variation is caused by the Hamming distance between current and previous data values in RTL (register transfer level) behavior. There are commercial microprocessors designed in static CMOS, but we can exploit many other advantages in terms of energy as well as speed with dynamic CMOS, due to its superior performance in terms of spurious transitions, short-circuit currents and parasitic capacitances [15], [16]. In particular, the datapath components, are largely dynamic in high-performance design [17]. Lots of microprocessors, including ARM7TDMI, use dynamic CMOS logic. The *precharge-and-evaluation* scheme of dynamic CMOS logic consumes energy proportional to the number of 1s (or of 0s, depending on the circuit structure). Thus the energy characterization must be performed in terms of the number of 1s (or 0s) as well as in terms of the Hamming distance.

III. Real-Time Cycle-Accurate Energy Consumption Measurement

A. Principle of Operation

Instruction-level energy characterization is suitable for high level or software energy reduction. Most commercially available microprocessor-based systems are synchronous state machines. Therefore, state is a useful abstraction level for microprocessor-based systems, and thus the clock cycle is a useful basis for the measurement of energy consumption. First, we measure cycle-accurate energy consumption and then compose the energy profiles into an instruction-level energy characterization.

Conventional energy measurement relies on instrumentation of the voltage across a series resistor in the power supply line. The power spectrum of the voltage across the resistor is dominant up to $1/2t_f$, where t_f is the shortest fall time of the signal and is often 2 ns or less [7]. Thus one must sample the voltage at a very high rate for reasonable accuracy. This is a serious problem in terms of both analysis and measurement time, for systems of reasonable complexity.

In our approach, we measure the cycle-accurate energy consumption of synchronous state machines by instrumenting charge transfer using switched capacitors, as shown in Fig. 1. The switch pairs (connected by dashed lines) repeat on/off actions, alternately. The capacitors C_{S1} and C_{S2} ($C_{S1} = C_{S2}$) are charged with V_s during a clock cycle and discharged during the next cycle, powering the target processor. The energy initially stored in the capacitor C_{Si} is $(1/2)C_{Si}V_{Ci}^2$. We measure the remaining energy stored in C_{Si} with the final voltage of C_{Si}. Since CMOS synchronous state machines do not consume energy when they are stable, we can measure the voltage of C_{Si} free from spiky noise. Therefore, the switched capacitor method is robust to dynamic change in the power supply current. In real VLSI implementation, there is usually on-chip bypass capacitor that minimizes power supply line fluctuation, and this may cause measurement error (Fig. 2). We measure the exact capacitance of C_{Si}, whose capacitance is nano-farad order, with a precise capacitance meter. According to the electric charge conservation law, we calculate the on-chip bypass capacitance C_B as follows:

$$C_B = \frac{C_{S1}V_{C1}(i-) - C_{S1}V_{C1}(i+)}{V_{C1}(i+) - V_{C2}(i-)} \quad (1)$$

because $V_B(i-) = V_{C2}(i-)$ and $V_B(i+) = V_{C1}(i+)$. The energy consumption for a clock cycle $clk(i)$, denoted by $E(i)$, is given by

$$E(i) = \tfrac{1}{2}(C_{S1} + C_B)V_{C1}(i+)^2 - \tfrac{1}{2}(C_{S1} + C_B)V_{C1}(i++)^2. \quad (2)$$

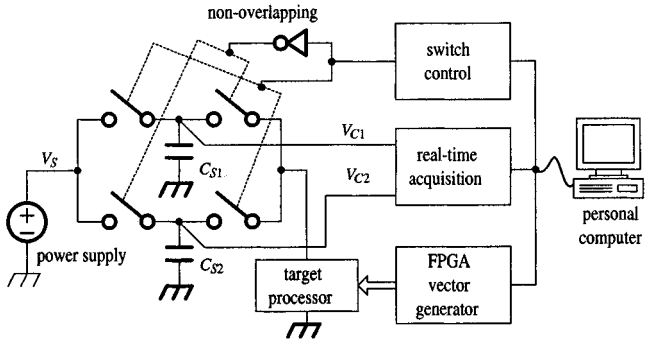

Fig. 1. Real-time cycle-accurate energy measurement system.

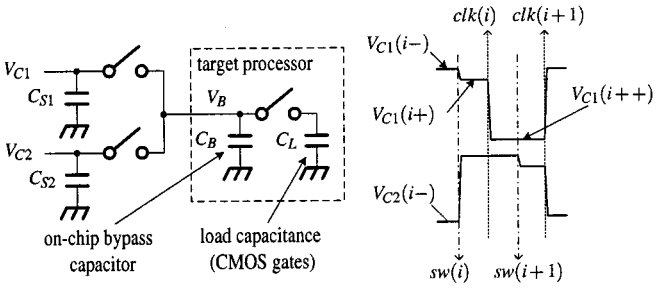

Fig. 2. Effect of on-chip bypass capacitor.

Fig. 1 illustrates the measurement system. We use CMOS bus switches (FST3125) which have a few nano-seconds switching time and several ohms on-resistance; lower on-resistance is achieved by parallel connections. The voltage of the capacitors, V_{C1} and V_{C2}, are converted by a 10-bit differential-input pipelined A/D converter (ADS826). The system stores the energy consumption profile in a high-speed SRAM in real time. An FPGA vector generator minimizes the interference added by the measurement process. For instance, it allows a microprocessor to execute repeatedly without adding jump or loop instructions.

The real-time acquisition unit samples both the control and the address signals to associate the energy value with each instruction. We can measure the exact energy consumed for a clock cycle of CMOS circuits with two sampling points because V_{C1} and V_{C2} remain stable when the circuit becomes stable, finishing the transition propagation. Fig. 3 shows a photograph of our in-house measurement tool.

In order to verify the accuracy of the measurement system, we also measure the average power supply current with a true RMS digital multimeter as we made infinite loops with the target instructions. We convert the current value into unit energy for a clock cycle and compare with our cycle-accurate energy measurement as shown in Table I.

B. Energy Consumption of CMOS Microprocessors

Energy consumption of low-power CMOS microprocessors is mostly dynamic, while recent high-performance microprocessors consume significant leakage energy. Switching activity causes dynamic energy consumption of CMOS circuits. The switching activity is largely dependent on the Hamming distance of data between current and previous clock cycles in the static CMOS circuits. In this paper, we will call it *Hamming-distance-dependent dynamic (HDD) energy*. Each dynamic CMOS circuit precharges before every evaluation. It draws large current if it has been discharged in the previous evaluation (usually the previous clock cycle), but only a small current if the circuit has not been discharged. The energy consumption is mainly proportional to the weight of the current data, i.e., the number of 1s (or 0s depending on the circuit structure). In this paper, we call

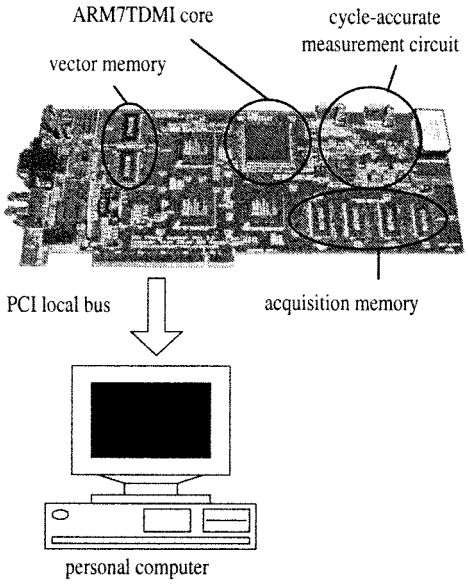

Fig. 3. Our in-house measurement tool.

TABLE I
MEASUREMENT ERROR OF THE CYCLE-ACCURATE MEASUREMENT SYSTEM (ENERGY: nJ)

Instruction	add	and	orr	sub	tst	mov	cmp	bic
DMM	1.29	1.16	1.36	0.86	1.16	1.35	0.85	0.85
Ours	1.32	1.19	1.39	0.88	1.19	1.38	0.87	0.87
Err (%)	2.21	2.47	2.28	2.81	2.69	1.99	2.36	2.42

it *weight-dependent dynamic (WDD) energy*. The WDD energy has been ignored in previous energy characterization, but its contribution is significant.

We characterize the variation of the HDD and WDD energy in terms of the instruction-level energy-sensitive factors. The characteristics of the HDD and WDD energy variations do not fully describe the total energy consumption: the energy consumption must be represented by the summation of the HDD, WDD, *common-mode dynamic* and *leakage* energies. We let the common-mode dynamic energy be the minimum value of the HDD and WDD energy; we cannot reduce this energy with high-level or software techniques. Therefore, the HDD energy and WDD energy represent the amount that may be reduced by high-level techniques.

Relative power consumption is more important in RTL-energy energy reduction [18]. Dynamic energy consumption takes place in CMOS circuits only when the leakage energy is negligible. It is important to reduce the leakage energy in low-level low-power techniques; but high-level techniques do not achieve reduction of leakage beyond that obtained by operating the devices properly. In this paper, we are aiming for high-level or software-level reduction, and thus we describe the variation of energy consumption in terms of the HDD and WDD, which are dependent in turn on the Hamming distance and the number of 1s.

C. Energy Measurement of Pipelined Microprocessors

The CPI (clock cycle per instruction ratio) of modern microprocessors is near to 1, due to pipelined operation. The instruction-level energy consumption is the summation of energy consumed at each pipeline stage. Cycle-accurate energy measurement tells us the energy consumption of a clock cycle, which is the total energy of all the pipeline stages.

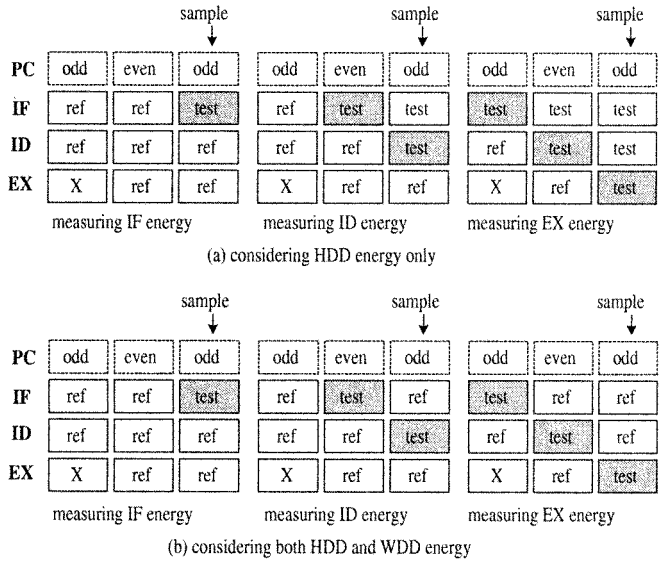

Fig. 4. Pipeline setup for measuring each pipeline stage (PC independent).

We measure the energy variation with various $(ref,\ test)$ instruction pairs. The HDD energy is dependent on the Hamming distance between the $(ref,\ test)$ instruction pairs, and the WDD energy is dependent on the weight of the $test$ instruction. The key idea is to measure the energy variation at each pipeline stage by changing the $test$ instruction while keeping the other sources of energy consumption constant. Fig. 4 illustrates pipeline setups to measure the energy variation. If the target processor consumes only the HDD energy, the pipeline setup will be trivial, as shown in Fig. 4(a). However, we need a somewhat elaborate pipeline setup because the ARM7TDMI consumes both HDD and WDD energy. We use both pipeline setups as shown Fig. 4(a) and (b). There is no explicit PC stage in the ARM7TDMI, but the instruction fetch address is issued at Phase 2 of the previous cycle, and thus we distinguished the PC stage from the IF stage during both measurement and analysis. We cannot independently measure energy variations due to opcode encoding and operations for COTS microprocessors, but we try to distinguish energy variations between the opcode encoding and operations, in order to improve our characterization.

The pipeline setup for the PC stage is the simplest. We can change the instruction fetch address by performing address relocation of the code without disturbing other pipeline stages. The IF stage is also straightforward. We can freely exchange the $test$ instruction with a given ref instruction while maintaining the Hamming distance between the $(ref,\ test)$ instructions and the weight of the ref instructions at the other pipeline stages. The EX and ID stages have fewer degrees of freedom. We chose a ref instruction for the ID stage, keeping the same base cost in the EX stages or compensating for a change in base cost. Changes in Hamming distance between the instruction fetch address values can also produce bias, so we locate the measurement points at the even addresses (PC stage) to keep the Hamming distance to 1.

D. Experimental Setup for ARM7TDMI

Our target processor is an ARM7TDMI [19] test chip that is manufactured by Hynix Co., Ltd.[1] It embeds only the ARM7TDMI CPU core and the ICEBreaker. It does not contain other peripherals, unlike common ARM7TDMI-based embedded controllers, and is thus

[1]All the measurement results are for the ARM7TDMI from Hynix Co., Ltd., which exhibits slightly larger energy consumption and variation than the ARM7TDMI from Seiko Epson Co. Ltd. that is shown in [20].

suitable for energy measurement. This chip also has separate power supply pins for the processor core, so we can easily decouple side-effects caused by differently loaded I/O pins. Conventional processor boards may have differently loaded memory buses, even though the target processors are the same; this may result in the measurement of energy variation, which is in fact largely caused by the bus and peripherals rather than by the processor. In this case, each instruction may show a distinct average power consumption with small variance. The experiment may appear successful, but the measured data exaggerates the effect of instruction encoding and the fetch addresses. Our in-house tool is designed to minimize the bus effect by using bus switches, and an FPGA vector generator in case the target processor does not have separate power supply pins. The address, the data, and all the control pins are connected to the FPGA vector generator, which is able to control the target processor with considerable flexibility.

We cross-compile ARM7 programs for proper pipeline setups and download the binary image to the FPGA vector generator. Thus it is simple to upload the energy consumption profile from the measurement system.

IV. Energy Measurement of the ARM7TDMI Core

A. PC Stage

We measure the PC-stage energy by changing the address location of various $(ref,\ test)$ instruction pairs. Energy variation is proportional to the Hamming distance between the instruction fetch addresses of the $(ref,\ test)$ instruction pairs, with a maximum variation of 224pJ, as shown in Fig. 5.

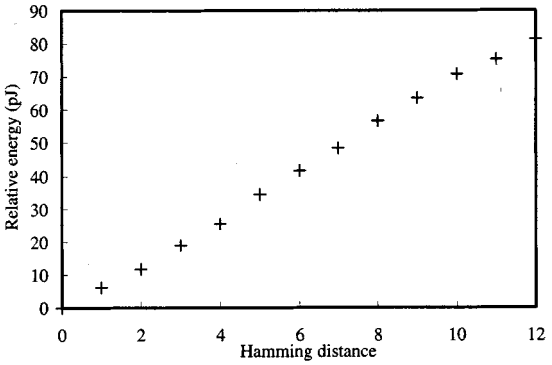

Fig. 5. Energy consumption against instruction fetch address (PC stage).

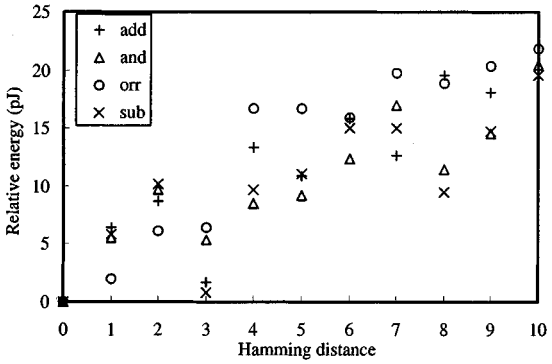

Fig. 6. Energy consumption against opcode encoding (IF stage).

B. IF Stage

We measure the IF-stage energy variation by changing the energy-sensitive factors of the $(ref,\ test)$ instruction pairs, using four different ref instructions. The Hamming distance between opcodes, register numbers and immediate operands of the $(ref,\ test)$ instruction pairs causes major energy variation, but we observe only the HDD energy. No significant difference between $(ref,\ test)$ instruction pairs is observed as long as the Hamming distances are the same; their energy variations are orthogonal to the operation.

Fig. 6 shows energy variation against Hamming distance for the opcode fields in $(ref,\ test)$ instruction pairs. Fig. 7 illustrates the energy variation due to the Hamming distance between the register numbers in $(ref,\ test)$ instruction pairs. Fig. 8 shows that the Hamming distance between the immediate operand fields affects the IF-stage energy. The total IF-stage energy variation is relatively small, except for the immediate operand field, although distinct energy variation is confirmed.

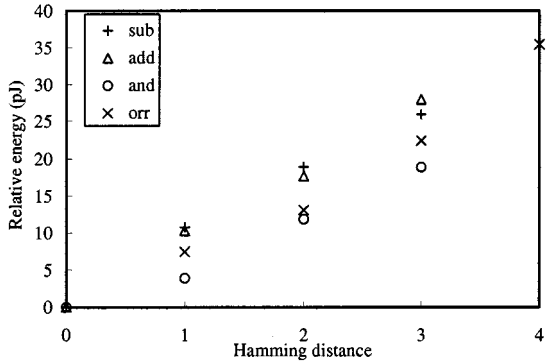

Fig. 7. Energy consumption against register number (IF stage).

C. ID Stage

We observe that the ID-stage energy varies with the register numbers, immediate operand values and register values of $(ref,\ test)$ instruction pairs; we observe both the HDD and WDD energy.

The variation in HDD energy is caused by the register numbers and the immediate operand value. Fig. 9 shows that the ID-stage energy is proportional to the Hamming distance between the register number fields of the $(ref,\ test)$ instruction pairs. Fig. 10 shows the effect of the immediate operand values at the ID stage: it is proportional to the Hamming distance between the $(ref,\ test)$ instruction pairs. These characteristics are qualitatively the same as those of the IF stage, although their magnitude is even larger.

We observe that each operation of a $test$ instruction causes unique energy variations. These are categorized as WDD energy because the unique value of each $test$ instruction does not vary with the ref instruction. Fig. 11 shows the energy variation of various $test$ instructions with four different ref instructions. These values result in *instruction base costs*. In fact, the base costs are less useful than other energy variations in high-level power reduction because we have fewer alternatives to explore.

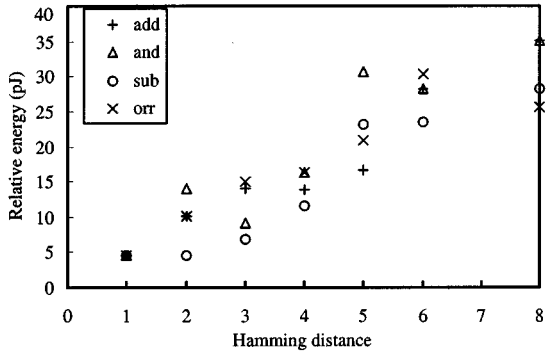

Fig. 8. Energy consumption against immediate operand value (IF stage).

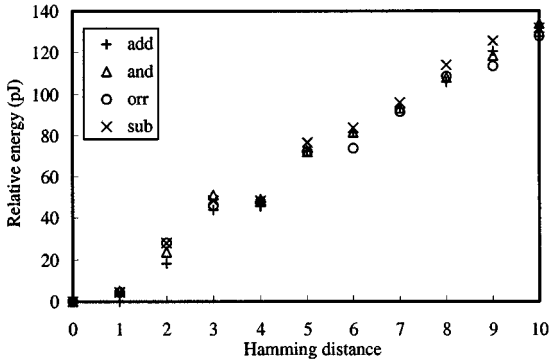

Fig. 9. Energy consumption against register number (ID stage).

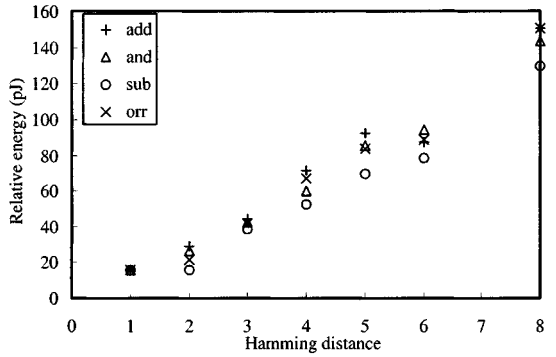

Fig. 10. Energy consumption against immediate operand value (ID stage).

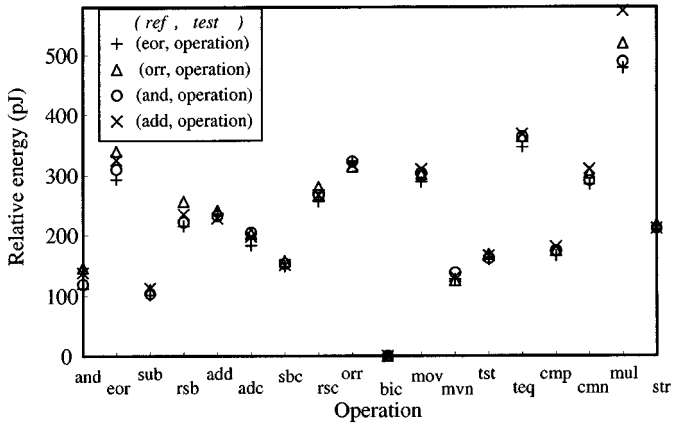

Fig. 11. Energy consumption against operation (ID stage).

The ID-stage energy is significantly affected by the register values. The energy is proportional to the number of 1s in the *A bus* and proportional or inversely proportional to the number of 1s in the *B bus*, as shown in Figs. 12 and 13.

D. EX Stage

We observe that the EX-stage energy is dependent on the register numbers, the register values, the immediate operand values and the choice of operations. There is both the HDD energy and WDD energy.

First, we measure the energy variations due to differing register values over 11 instructions, and found that the energy consumption is proportional to the number of 1s in the values. Fig. 14 illustrates that the energy consumption is largely independent of the type of operation: the variation is no more than 550 pJ. Secondly, we measure

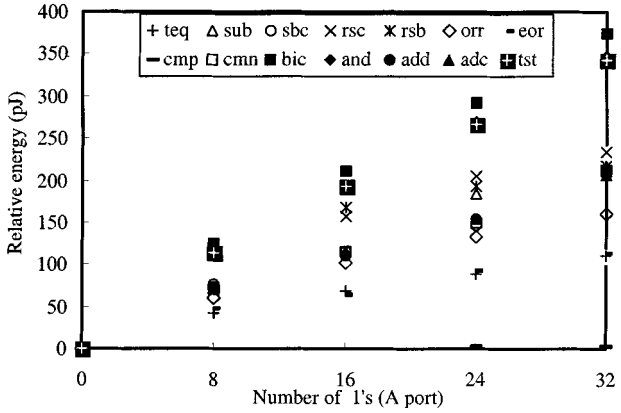

Fig. 12. Energy consumption against register value (A-port registers, ID stage).

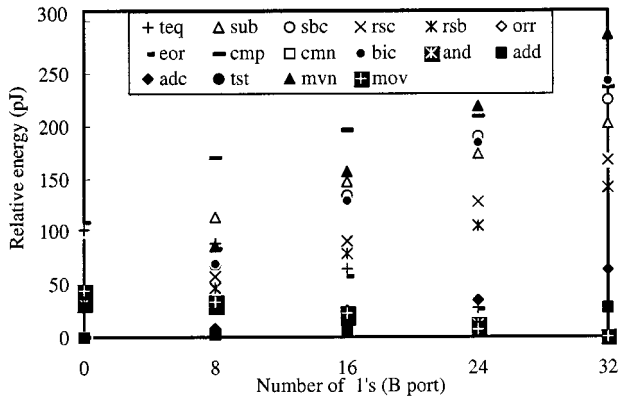

Fig. 13. Energy consumption against register value (B-port registers, ID stage).

how the energy varies with register number. Fig. 15 shows that the EX-stage energy is proportional to the Hamming distance between the register numbers in $(ref, test)$ instruction pairs. Thirdly, we confirm that the immediate operand value also affects the EX-stage energy. The trend is similar to the register values, as shown in Fig. 16.

Finally, we measure the EX-stage energy for each operation by keeping other factors the same while issuing four different ref instructions. The operation of the ref instruction does not affect the unique base cost associated with each operation. The total variation with opcode is significant, as shown in Fig. 17. These results also provide the instruction base costs.

E. Multicycle Instructions

Multicycle instructions occupy more than two EX-stage cycles while causing other stages to stall. Fig. 17 shows the base cost of `str` and `mul` instructions for the first, middle (one or more) and last cycles. The first EX cycle of a `str`/`ldr` (`rs1`), `rs2` instruction transfers the effective memory address to the *address register* and the energy variation agrees with Fig. 14 for corresponding values of `rs1` and `rs2`. The next EX stage performs a memory operation, and the energy variation confirms the results shown in Fig. 5, for varying values of Hamming distance between the effective address value and the ref instruction fetch address. The number of EX cycles of the `mul` instruction is dependent on the data (and is computed using the Booth algorithm). The EX-stage energy of `mul` is not significantly variable; but what variation there is, does not agree with Fig. 14.

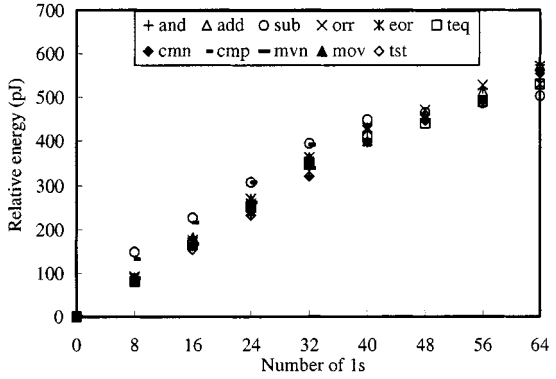

Fig. 14. Energy consumption by the operand value (EX stage).

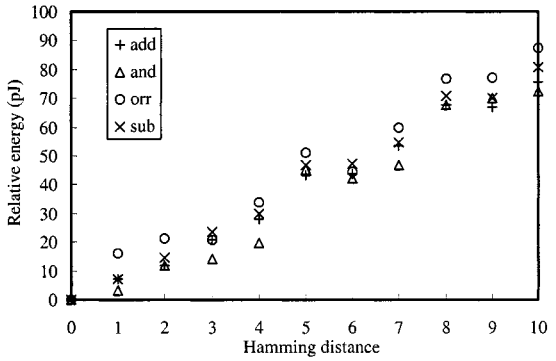

Fig. 15. Energy consumption by the register number (EX stage).

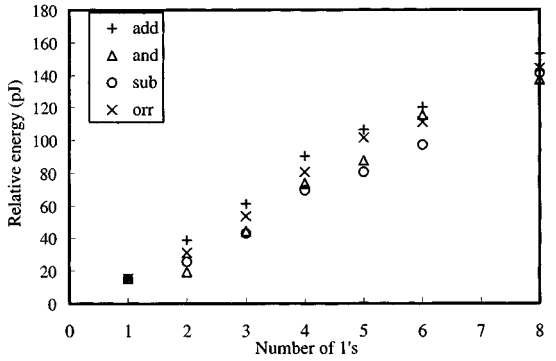

Fig. 16. Energy consumption by the immediate value (EX stage).

V. Energy Characterization of the ARM7TDMI Core

A. Two-Dimensional Characterization

Previous authors [8], [5] have introduced instruction-level characterizations in the form of instruction base cost and inter-instruction cost.

The base and inter-instruction costs form a two-dimensional (2-D) table as shown in Table II; in this paper we will call it a *2-D characterization*. The base costs $\overline{b}_j$ and inter-instruction costs $\overline{i}_{i,j}$ average out the energy variations caused by most instruction-level energy-sensitive factors. As demonstrated in Section IV, the energy variations are in general orthogonal to the operations. Therefore, most important energy characteristics escape this characterization; moreover, the lost energy variations are the most significant.

The 2-D characterization may do no more than encourage low-energy software designers to avoid using expensive instructions or bad combinations, which is a restrictive approach. Consequently, the 2-D characterization is suitable for average energy estimation, and does not furnish useful information for software energy reduction.

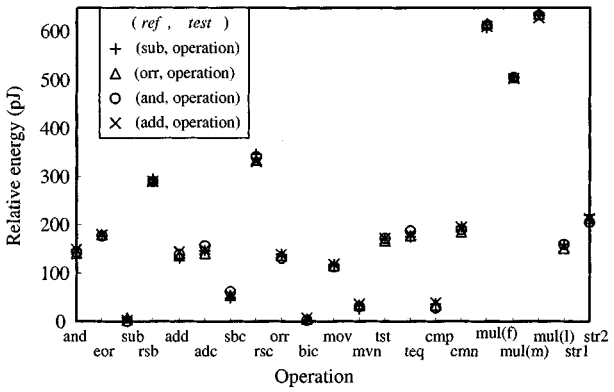

Fig. 17. Energy consumption against opcode (EX stage).

TABLE II
2-D Characterization of Instruction Costs

	instr_1	instr_2	instr_3	instr_4	instr_n
instr_1	$\overline{b}_1$	$\overline{i}_{1,2}$	$\overline{i}_{1,3}$	$\overline{i}_{1,4}$	$\overline{i}_{1,n}$
instr_2	$\overline{i}_{2,1}$	$\overline{b}_2$	$\overline{i}_{2,3}$	$\overline{i}_{2,4}$	$\overline{i}_{2,n}$
instr_3	$\overline{i}_{3,1}$	$\overline{i}_{3,2}$	$\overline{b}_3$	$\overline{i}_{3,4}$	$\overline{i}_{3,n}$
instr_4	$\overline{i}_{4,1}$	$\overline{i}_{4,2}$	$\overline{i}_{4,3}$	$\overline{b}_4$	$\overline{i}_{4,n}$
instr_n	$\overline{i}_{n,1}$	$\overline{i}_{n,2}$	$\overline{i}_{n,3}$	$\overline{i}_{n,4}$	$\overline{b}_n$

B. Multidimensional Characterization

There are some limited analysis results [5] which relate to some of the energy-sensitive factors, such as average energy variation due to addressing modes and data bus activities. Operand-dependent energy analysis has also been applied [1] to some representative components.

In this paper, we propose to use a *multidimensional characterization* that preserves the energy characteristics of all the significant instruction-level energy-sensitive factors. (We are not interested in factors which are not controllable by software, even though they affect energy consumption by processor.) When we average out and simplify energy variation related to instruction fetch addresses, immediate operand values, register numbers and register values, this multidimensional characterization will be identical to the inter-instruction characterization. Section IV describes the variation of HDD and WDD energy with respect to these factors. We summarize effective energy-sensitive factors of each pipeline stage in Fig. 18.

Definition 1: The set of effective energy-sensitive factors of a microprocessor, Ξ, is a seven-tuple $(\phi, \varepsilon, \delta, \rho, \mu, \omega, \iota)$ where

- ϕ: instruction fetch address;
- ε: opcode encoding;
- δ: operation;
- ρ: register number;
- μ: memory address;
- ω: register value;
- ι: immediate operand.

■

We define a functions $f_h(x, y)$, $x, y \in \Xi$, that denotes the Hamming distance between x and y, and a function $f_w(x)$, $x, \in \Xi$, that defines the number of 1s of x. We denote the energy-sensitive factors of instructions i and j as Ξ_i and Ξ_j, respectively. Most energy characteristics associated with ξ are almost linear. We characterize each graph as a first-order equation with simple regression analysis, as shown in Table III.

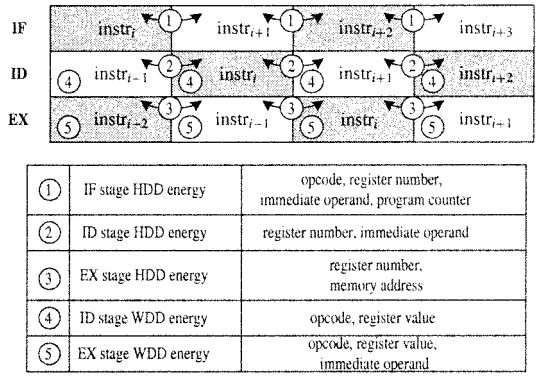

Fig. 18. Components in a programmers' model and their energy contributions.

So far, we have classified energy consumption into HDD, WDD, common-mode and leakage energies. Now, we relate the energy characteristics to instruction-level characterization. We divide Ξ into *operation-dependent* and *operation-orthogonal* parts. The operation-dependent part for the ARM7TDMI includes δ and ω at the ID stage, as shown in Section IV. The energy consumption of instruction j following i is given by:

$$E(j|i) = \sum_{\xi_i,\, \xi_j \in (\Xi - \{\delta\})} (\alpha_\xi f_h(\xi_i,\, \xi_j) + \beta_\xi f_w(\xi_j)) + f_\delta(\delta_j) + f_{\omega|id}(\delta_j) f_w(\omega_j) + \eta. \quad (3)$$

The first term of the right-hand side of (3) is operation-orthogonal, and the second and third terms are operation-dependent; the last term η is the common-mode base cost. The ARM7TDMI core does not have a distinct leakage energy. We find the coefficients α_ξ and β_ξ, $\xi \in \Xi$, and also the functions $f_\delta(\delta)$ and $f_{\omega|id}(\delta)$ by regression analysis of the characteristic graphs presented in Section IV. Table IV shows the coefficients α_ξ and β_ξ and supplies useful information for various low-energy software techniques including register re-encoding and instruction rescheduling.

Table V shows the operation-dependent base costs, $f_\delta(\delta)$. We obtain an 792 pJ common-mode base cost, η, for the ARM7TDMI test chip. All the values are measured while $f_h(\xi_i,\, \xi_j) = f_w(\xi_j) = 0$, where $ref = i$, $test = j$ and $\xi_i,\, \xi_j \in \Xi$. The function $f_{\omega|id}(\delta)$ describes the operation-dependent energy variation produced by changing register values at the ID stage. This is difficult to characterize because the A-bus energy and B-bus energy are cross-coupled. To compose Table VI we take average slopes. Even if we were to characterize $f_{\omega|id}(\delta)$ in much more complex ways, we would still have limitations in using $f_{\omega|id}(\delta) f_w(\omega)$ in real-world energy reduction practices, because it is simultaneously dependent on the data, $f_w(\omega)$, and the operation, $f_{\omega|id}(\delta)$.

The relation between conventional base cost, $\overline{b}_j$, is denoted as follows:

$$\overline{b}_j = f_\delta(\delta_j) + \overline{\sum_{\xi_i,\, \xi_j \in (\Xi - \{\delta\})} (\alpha_\xi f_h(\xi_i,\, \xi_j) + \beta_\xi f_w(\xi_j))} + f_{\omega|id}(\delta_j)\overline{f_w(\omega_j)} + \eta. \quad (4)$$

The inter-instruction cost, $\overline{i}_{i,\,j}$, is denoted by

$$\overline{i}_{i,\,j} = f_h(\varepsilon_j,\, \varepsilon_j). \quad (5)$$

The term $\overline{i}_{i,\,j}$ reflects the effect of ε, which mainly influences the IF-stage energy variation and accounts for less than 2.1% of the total variation. Other factors orthogonal to the operations are much more significant.

TABLE III
RELATIVE ENERGY CONSUMPTION MODEL OF INSTRUCTION j FOLLOWING INSTRUCTION i (ARM7TDMI TEST CHIP, pJ)

Ξ	IF stage		ID stage		EX stage	
	E (pJ)	%	E (pJ)	%	E (pJ)	%
ϕ	$7.0 f_h(\phi_i,\phi_j)$	14.23	0	0	0	0
ε	$8.3 f_h(\varepsilon_i,\varepsilon_j)$	2.11	0	0	0	0
ρ	$1.9 f_h(\rho_i,\rho_j)$	1.45	$13.6 f_h(\rho_i,\rho_j)$	10.37	$8.2 f_h(\rho_i,\rho_j)$	6.25
μ	0	0	Figures 12 and 13		$7.0 f_h(\mu_i,\mu_j)$	14.23
ω	0	0	Figures 12 and 13		$8.8 f_w(\omega_j)$	35.78
ι	$4.3 f_h(\iota_i,\iota_j)$	2.19	$18.1 f_h(\iota_i,\iota_j)$	9.20	$19.0 f_w(\iota_j)$	9.66

TABLE IV
OPERATION-ORTHOGONAL COEFFICIENTS, α_ξ AND β_ξ, WHERE $\xi \in \Xi$ (ARM7TDMI TEST CHIP, pJ)

α_ξ: Hamming-distance coefficient, β_ξ: Weight coefficient

coefficient	α_ϕ	α_ε	α_ρ	α_μ	α_ω	α_ι
value	7.0	8.3	23.7	7.0	0.0	22.4
coefficient	β_ϕ	β_ε	β_ρ	β_μ	β_ω	β_ι
value	0.0	0.0	0.0	0.0	8.8	19.0

TABLE V
OPERATION-DEPENDENT BASE COST FUNCTION, $f_\delta(\delta)$ (ARM7TDMI TEST CHIP, pJ)

operation (ε)	$f_o(\varepsilon)$	operation (ε)	$f_o(\varepsilon)$	operation (ε)	$f_o(\varepsilon)$
and	275	eor	496	sub	111
rsb	526	add	372	adc	346
sbc	206	rsc	606	orr	456
bic	0	mov	418	mvn	161
tst	335	teq	540	cmp	208
cmn	489	mul	2267	str	582

TABLE VI
OPERATION-DEPENDENT REGISTER VALUE VERSUS ID-STAGE ENERGY COEFFICIENT FUNCTION, $f_{\omega|id}(\delta)$ (ARM7TDMI TEST CHIP, pJ)

operation (ε)	$f_c(\varepsilon)$	operation (ε)	$f_c(\varepsilon)$	operation (ε)	$f_c(\varepsilon)$
and	4.79	eor	-0.02	sub	5.98
rsb	5.29	add	3.53	adc	4.04
sbc	6.65	rsc	6.30	orr	1.87
bic	9.49	mov	-1.44	mvn	8.83
tst	4.76	teq	0.02	cmp	3.28
cmn	3.53	mul	–	str	–

VI. APPLICATIONS

A. Energy Scope and Function-Level Characterization

It used to be difficult to measure energy variation at the clock-cycle level with conventional equipment. Fig. 19 shows the cycle-accurate energy consumption of a popular IDCT row function. We observe that the ARM7TDMI core consumes a significantly different amount of energy, depending on the number of 1s in the input data. We use four different types of IDCT input data for function-level characterization: each consists of eight 32-bit words and contain the same numbers of

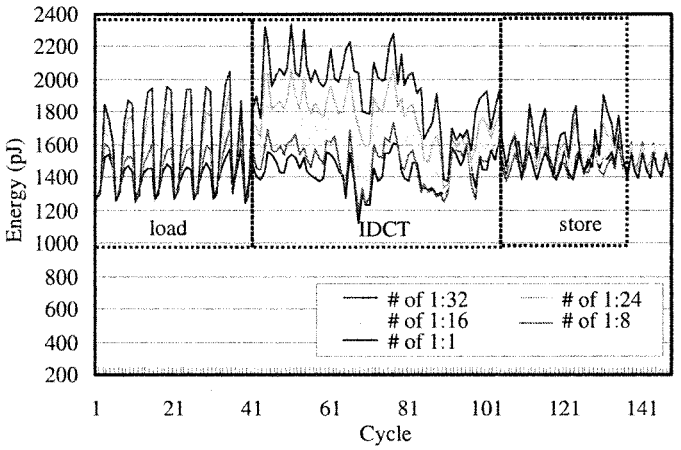

Fig. 19. Energy consumption of IDCT routine with different input data (Type 1).

1s, but their values are different. Fig. 20 shows the relative energy variation of the IDCT function with the input data.

B. Software Energy Reduction

The energy characterization presented in Section V motivates various software power reduction schemes. As long as compliers have no knowledge about energy consumption, they are frequently likely to generate energy-inefficient codes by accident. In this section, we describe two energy reduction techniques that employ the IDCT function.

The energy characteristics show that the instruction fetch energy can be optimized by reducing the Hamming distance between the address values. We can also see that the reduction will be 7.0 pJ per bit of Hamming distance. Fig. 21 shows the energy consumption of the IDCT function with different input data segment values. The code segment is `0x00 000 000`, and the text is located from `0x00 000 000` to `0x00 000 230`. We use two data segment values, `0x00 000 400` and `0x00FFFC00`. The energy difference is 5.7% for load, 7.7% for store, and 3.5% overall. The load operation shows a smaller reduction than the store instruction because the `ldr` instruction takes one more cycle than the `str` instruction.

The register number may change the energy consumption at the IF, ID and EX stages by up to 1.9%, 13.6% and 8.2%, respectively. With 30% Hamming distance reduction, we can achieve 7.2% reduction for the associated instructions. Fig. 22 shows the energy difference caused by register encoding. The original code, generated by an ARM7 compiler (ARM Software Development Toolkit v 2.02u) has an 898-bit Hamming distance among the register number fields. It generates the same code for the IDCT function regardless of the optimization options. We changed the register encoding and obtained Hamming distances of 797 bits in the best case, and 995 bits in the worst. We can find some adjacent instructions that have longer Hamming distances than the original code, but the optimal encoding has a shorter overall Hamming distance and results in a 1.8% energy reduction for the whole routine.

A simple calculation shows that the power consumption of the same instruction may differ by more than 100%. There seem to be no one scheme that dramatically reduces the energy consumption, but several techniques together may achieve a satisfactory result.

VII. Conclusion

Measurement-based energy characterization has many advantages over limited conventional techniques in real-world low-power design. In this paper, we have developed a real-time cycle-accurate energy

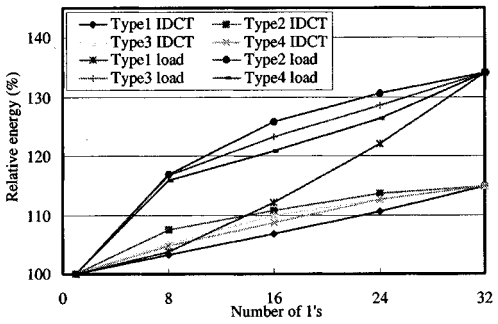

Fig. 20. Function-level energy characterization of the IDCT row function against input data.

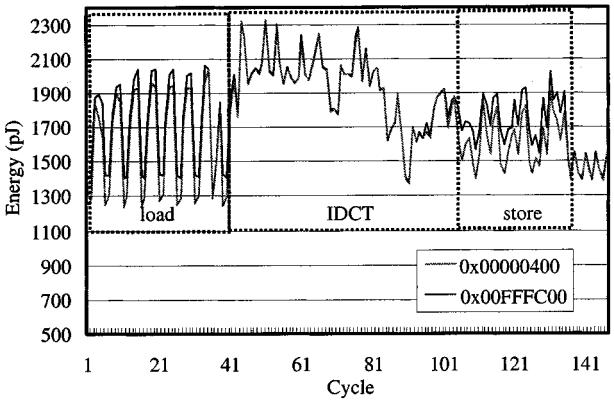

Fig. 21. Energy consumption of the IDCT routine with different data pointer values.

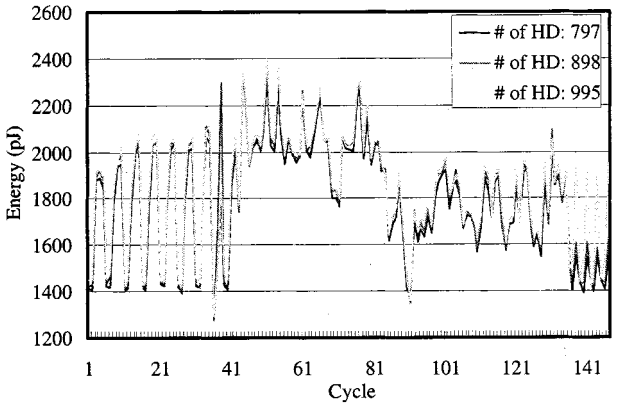

Fig. 22. Energy consumption of the IDCT routine with different register encoding.

measurement method that is based on the instrumentation of charge transfer. This technique enables us to characterize various COTS digital systems even faster than by simulation.

So far, energy reduction has largely been studied for artificial systems, with lots of assumptions. We have shown the actual energy behavior of a COTS microprocessor though a case study. We measured the energy variations of the ARM7TDMI core in terms of opcodes (operations), register numbers, register values, instruction fetch addresses, data fetch addresses, and immediate operand values at each pipeline stage, and composed them into instruction-level characterization.

In this paper, we characterized a 32-bit RISC microprocessor and introduced substantial energy reduction guidelines. First, we have not averaged out the energy characteristics, unlike previous characterizations.

Secondly, we directly measured a real microprocessor and thus demonstrated quantitative analysis of energy consumption. Thirdly, we distinguish energy variations caused by energy-sensitive factors from each pipeline stage. Finally, we observed strong energy variation caused by the number of 1s as well as the Hamming distance in dynamic CMOS circuits, which are common in low-power, high-performance microprocessors.

Measurement-based characterization reflects real implementation. We have demonstrated significant energy variation due to the number of 1s in RTL behavior, which is caused by dynamic CMOS logic and has usually been ignored in earlier characterizations.

REFERENCES

[1] D. Sarta, D. Trifone, and G. Ascia, "A data dependent approach to instruction level power estimation," in *Proc. IEEE Alessandro Volta Memorial Workshop . Low-Power Design*, Mar. 1999, pp. 182–190.

[2] R. Yu Chen, M. J. Irwin, and R. S. Bajwa, "An architectural level power estimator," *Proc. ISCAW*, June 1998.

[3] R. Yu Chen, R. M. Owens, M. J. Irwin, and R. S. Bajwa, "Validation of an architectural level power analysis techinque," in *Proc. 35th Design Automation Conf.*, June 1998, pp. 242–245.

[4] T. Simunic, L. Benini, and G. De Micheli, "Cycle-accurate simulation of energy consumption in embedded systems," in *Proc. 36th Design Automation Conf.*, June 1999, pp. 867–872.

[5] P. Laramie, "Instruction level power analysis and low power design methodology of a microprocessor," Master thesis, Dept. of Electrical Engineering and Computer Sciences, U. C. Berkeley, 1998.

[6] W. Ye, N. Vijaykrishnan, M. Kandermir, and M. J. Irwin, "The design and use of SimplePower: A cycle-accurate energy estimation tool," in *Proc. Design Automation Conf. 2000*, June 2000, pp. 340–345.

[7] H. W. Johnson and M. Graham, *High-Speed Digital Design a Hand Book of Black Magic*. Englewood Cliffs, NJ: Prentice-Hall , 1993.

[8] V. Tiwary, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step toward software power minimization," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 437–445, Dec. 1994.

[9] M. Tien-Chien, V. Tiwari, S. Malik, and M. Fujita, "Power analysis and low-power scheduling techniques for embedded DSP software," in *Proc. 8th Int. Symp. Syst. Synthesis*, Sept. 1995, pp. 110–115.

[10] J. Russel and M. Jacone, "Software power estimation and optimization for high perfomance, 32-bit embedded processors," in *Proc. Int. Conf. Comput. Design*, Oct. 1998, pp. 328–333.

[11] J. Bradley. (1993) Calculation of TMS320C5x power dissipation application report. [Online]. Available: http://www.ti.com/sc/docs/psheets/abstract/apps/spra030.htm

[12] T. Burd and B. Peters. (1994) A power analysis of a microprocessor: A study of an implementation of the MIPS R3000 architecture. ERL Technical Report. [Online]. Available: http://bwrc.eecs.berkeley.edu/burd/gpp

[13] J. Lorch and A. J. Smith, "Energy consumption of apple macintosh computers," *IEEE Micro*, vol. 18, pp. 54–63, November/December 1998.

[14] A. Wolfe, "Opportunities and obstacles in low-power system-level CAD," in *Proc. 33rd Annu. Conf. Design Automation Conf.*, June 1996, pp. 15–20.

[15] A. P. Chandrakasan, S. Shung, and R. W. Brodersen, "Low power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, pp. 473–484, Apr. 1992.

[16] E. Macii and M. Poncino, "Power consumption of static and dynamic CMOS circuits: A comparative study," in *Proc. Int. Conf. ASIC*, Oct. 1996, pp. 425–427.

[17] S. Rajgopal, "Challenges in low-power microprocessor design," in *Proc. Int. Conf. VLSI Design*, Jan. 1996, pp. 329–330.

[18] A. Bogiolo, L. Benini, and G. De Micheli, "Characterization-free behavioral power modeling," *Design, Automation and Test in Europe*, pp. 767–773, Feb. 1998.

[19] S. Furber, *ARM System Architecture*. Reading, MA: Addition-Wesley, 1997.

[20] N. Chang, K.-H. Kim, and H. G. Lee, "Cycle-accurate energy consumption measurement and analysis: Case study of ARM7TDMI," in *Proc. Int. Symp. Low Power Electron. Design*, July 2000, pp. 185–190.