# System-Level Power Estimation using an On-Chip Bus Performance Monitoring Unit *

Youngjin Cho, Younghyun Kim, Sangyoung Park and Naehyuck Chang
Dept. of EECS, Seoul National University
{yjcho, yhkim, sypark, naehyuck}@elpl.snu.ac.kr

## ABSTRACT

In this paper we propose an on-chip bus PMU which makes accurate estimates of system power consumption from a first-order linear power model by utilizing system-level activity information exchanged on the on-chip bus. It can easily be customized for different on-chip and off-chip memory devices, and is not dependent on a specific CPU core. We model memory devices using energy state machines, describe them in XML, and use that description automatic synthesis of the PMU. We compare the short-term accuracy of the proposed PMU with a cycle-accurate system-level power estimator, and assess its long-term accuracy with a real hardware prototype. Experimental results show that the the power estimation deviates less than 5% from real measurements.

## 1. INTRODUCTION

Power consumption has now become one of the most important considerations in designing digital systems, so system designers need to measure the power used by the system as well as its performance. In a laboratory environment, a range of use of various measurement equipment, and a specially designed prototyping board that allows current probing, can be used to determine detailed device- and system-level power consumption [1]. A more advanced charge measurement scheme can even report the energy consumption of each device during each cycle [2]. However, it is not economic to incorporate precise current, voltage or charge measurement devices into the final product, even if there is a specific requirement for online power measurement.

Indirect power measurement using a PMU can significantly reduce the cost of power measurement. PMUs are composed of set of counters (they are purely digital logic), and which makes them attractive for low-cost in-system measurement. A PMU can produce a reasonable estimate of the estimate of the power consumption of a digital system because power consumption is strongly dependent on the system activities. However, the accuracy of this method of power estimation requires the capture of all the relevant system activities and a correct model of the relationship between these activities and the power consumption.

Existing PMUs can achieve acceptable accuracy in estimating CPU power consumption. However, the current generation of PMUs do not explicitly measure external memory references. Instead, most PMU-based power estimation techniques rely on cache miss counts to estimate memory power consumption [3, 4, 5, 6]. This method has an average error of 70% in estimating the power consumption of a straightforward SDRAM memory [3]. But, this is not surprising because a single memory device can exhibit a power consumption which differ by an order of magnitude or more between different types of operations, such as a page-miss single read and a burst-mode read. This variability is exacerbated in the wide range of heterogeneous memory devices including on-chip SRAM, off-chip SDRAM (DDR or SDR), and NOR flash, which use completely different amounts of power. This make it infeasible to estimate memory power accurately simply by counting the number of cache misses. However, a poor estimate of memory power severely compromises overall power estimation because memory devices consume more power than the CPU in systems such as portable devices [7].

We tackle these problems by introducing a PMU which resides on the on-chip bus, and which aims to achieve accurate system-level power estimation especially by taking range of memory devices into account. To our knowledge, this is the first attempt to estimate system-level power consumption by means of an on-chip bus PMU. While conventional PMUs are not capable of recognizing the address map and different types of memory devices, our PMU obtains this information by snooping the on-chip bus signals. The PMU has internal state estimators for different memory devices, and performance counters to capture the number of state transitions and the amount of time that a memory remains in each state. The associated power estimation software is thus supplied with a complete energy annotation of all the states and transitions that take place in the memory devices. This data is obtained by cycle-accurate measurement. All the necessary information for PMU synthesis is presented as an XML (extended markup language) description. An SoC design is then based on predefined memory controller IP, and an XML description is used to synthesize an on-chip bus PMU.

We have performed extensive assessment of the accuracy of the proposed PMU. We verified the long-term average power consumption by measuring power consumption of a specially designed prototyping board running benchmark programs, and compared the measured value with the estimates from PMU. Then we verified the relative accuracy of measurement of short-term power variation, which cannot be detected by digital multimeters, using a cycle-accurate system-level energy simulator. The PMU achieves more than 95% accuracy.

## 2. ON-CHIP BUS PMU FOR POWER ESTIMATION

### 2.1 Transaction-based CPU PMUs

In an early example of the use of a PMU to estimate energy consumption [4], the number of active CPU cycles, active floating-point unit cycles, L2 cache references, and external memory ac-

*This work is partly supported by the Brain Korea 21 Project and ETRI SoC Industry Promotion Center, Human Resource Development Project for IT SoC Architect. The ICT at Seoul National University provides research facilities for this study.

978-1-4244-2820-5/08/$25.00 ©2008 IEEE

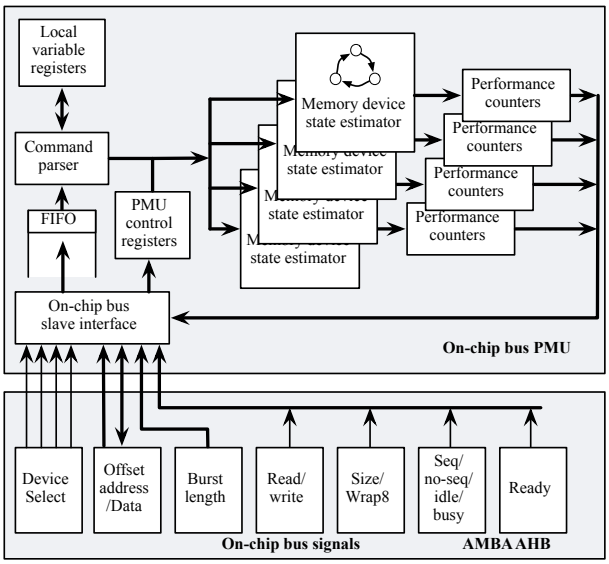


**Figure 1: Architecture of the proposed on-chip bus PMU.**

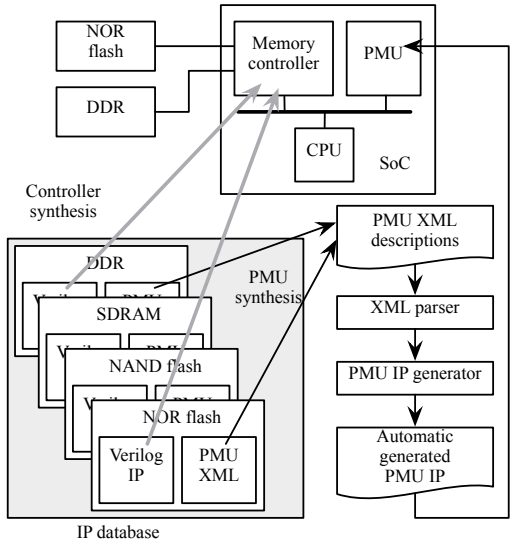


**Figure 2: PMU synthesis flow.**

cesses were counted by a PMU installed in a Pentium II microprocessor. Estimating power consumption from these counter values was shown to require a high level of abstraction, leading to elaborate parameter tuning. Later PMUs are basically similar to [4], but use more accurate methods of estimation. In a somewhat different approach [5], analytical energy models of each component of an UltraSPARC microprocessor including buses, memory cells and I/O pads were combined. Another technique [6] focused more closely on the CPU core and cache memory, giving more consideration to the microarchitecture. Linear regression has also been applied [8] to PMU-based energy estimation of a CPU core.

The accuracy of the tuned energy parameters has usually been verified by multimeter measurements [4, 6, 8]. An online feedback scheme [9] can be used to compensate for any error using energy consumption information from a battery management unit.

Conventional PMUs estimate the energy consumption of memory from the number of cache misses [4, 5, 6]; but this transaction-based approach is not very accurate. For example, a memory access initiated by the CPU or cache generally results in a single off-chip memory transaction. But one of those transactions often causes multiple off-chip transactions, because of misaligned incremental burst requests, burst requests that are too long to be supported by the memory devices, and other similar factors.

Sometimes, the accuracy of measurement may be compromised by the nature of the memory transactions. For instance, transaction-based PMUs cannot distinguish a block erase command from a normal data write in a flash memory, but these operations are very different, both in terms of energy consumption and execution time. Such memory devices have multiple internal states, and state transitions take place in response to the commands received. If the PMU treats an erase command as a normal data transfer, its estimates of power consumption will be wildly incorrect.

## 2.2 State-based on-chip bus PMUs

If a PMU is to understand the exact behavior of memory devices, it must be aware of the system memory map and the types of memory devices on the map, which is orthogonal to the CPU, and this places a basic restriction on the accuracy which can be expected from a CPU PMU.

This handicap can be overcome if the PMU is located on the on-chip bus, over which almost all the information necessary for system-level power estimation is exchanged. This information includes device select signals, device offset addresses, burst or single access indicators, burst lengths, read or write indicators. A more effective PMU for system-level performance and power estimation must be aware of the IP of the memory system components. This sort of PMU can also cooperate with an existing CPU PMU.

Figure 1 illustrates the proposed PMU architecture. Unlike the conventional transaction-based PMUs described in Section 2.1, it is composed of set of state machines that describe the internal behavior of each memory device. The PMU has a slave interface like a normal slave on-chip bus devices. As most modern memory devices accept pipelined memory transactions, the PMU also has a command FIFO queue. Once a particular memory device has been selected, the command parser triggers the associated state machine. The commands are encoded in the data bus and address bus to suit the particular memory devices. Other important signals such as read/write, burst length, size (word, halfword, etc.), sequential/non-sequential/idle/busy and ready are essential to help in correct parsing and to achieve correct state transitions. The parser stores useful values such as transfer size in the local variable registers. The PMU synthesis method will be described in Section 2.3.

The performance counters are a set of $N$ transition counters, denoted by $C[Device, T_i]$ $(1 \leq i \leq N)$, and a set of $M$ state counters, $C[Device, S_j]$ $(1 \leq j \leq M)$. Transition counters record how many transitions occur after a trigger (reset) for each transition $i$ defined in the state machine. State counters record the number of clock cycles during which the state machine stayed in each state $j$. $N$ is the number of transitions and $M$ is the number of states of the state machine.The area overhead of these counters is not significant in comparison with the size of the SoC; the area and power overhead is described in more detail in Section 3.3.

At the end of each time period, the CPU reads the counters in the PMU. The power consumption of each memory device is then calculated using the power estimation equations which will be described in Section 3.

## 2.3 PMU synthesis

To design the proposed on-chip bus PMU requires a level of knowledge about the memory devices similar to that needed to design a memory controller IP, because synthesis of the on-chip bus

**Table 1: Document type definition of the PMU XML**

```
<!ELEMENT PMU (Parameters?,Variables?,StateMachine+)>
<!ATTLIST PMU deviceType CDATA #REQUIRED>
<!ELEMENT Parameters EMPTY>
<!ATTLIST Parameters
                  onchip_bus_FIFO_depth (0|1|2|3|4|5|6|7|8)
                  counter_bitwidth CDATA "16"
                  onchip_bus_signal_latch (ON|OFF) "OFF">
<!ELEMENT Variables (Variable+)>
<!ELEMENT Variable EMPTY>
<!ATTLIST Variable
                  name CDATA #REQUIRED
                  type CDATA #REQUIRED
                  bitwidth CDATA "32"
                  initial CDATA "0">
<!ELEMENT StateMachine (State*)>
<!ELEMENT State (NextState*)>
<!ATTLIST State
                  name CDATA #REQUIRED
                  sid CDATA #IMPLIED
                  logging (ON|OFF) "ON">
<!ELEMENT NextState ( (Conditions | Automatic), Command?)>
<!ATTLIST NextState
                  did CDATA #IMPLIED
                  nextState CDATA #REQUIRED>
<!ELEMENT Conditions (#PCDATA)>
<!ELEMENT Automatic (#PCDATA)>
<!ATTLIST Automatic Unit (clk|ps|ns|us|ms|s) "clk">
<!ELEMENT Command (#PCDATA)>
<!ATTLIST Command when (TRUE|FALSE) "TRUE">
```

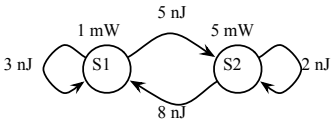


**Figure 3: Energy state machine example.**

PMU shares lots of design processes with synthesis of the memory controller IP. We added an XML description of the memory device state diagrams to the memory controller IP database as shown in Figure 2. During the memory controller design process, a designer can pick up the necessary memory controller IP from the IP design database for the memory devices being used. At the same time, the PMU generator creates the device state machine models from the XML descriptions in the IP database.

Device controller IP vendors or users can easily create an XML description from the controller hardware description or its simulation model. We follow the DTD (document type definitions) file as shown in Table 1. The root node tag, <PMU>, consists of variable declarations, parameters and state machine descriptions. The <Parameters> tag includes the options available in PMU synthesis. The <Variable> tag allows a user to declare the local variables for user-defined state machines. The <PMU> tag contains one or more <StateMachine> tags, and <StateMachine> consists of a series of <State> tags. Each <State> tag consists of a set of <NextState> tags, and each <NextState> tag has either a <Conditions> tag for a conditional state transition or an <Automatic> tag for autonomous state transitions which occur after a specified period. The PMU synthesis program regards consecutive <Conditions> and <Automatic> tags as OR-ed composite conditions. Optionally, it is possible to describe additional Verilog or SystemC subroutines using the <Command> tag.

# 3. PROPOSED METHOD OF POWER ESTIMATION

## 3.1 Energy state machine and cycle-accurate energy measurement

**Table 2: The characteristic parameters of the ARM926EJ-S microprocessor ($T_{pmu} = 1$ ms).**

| Parameter | Power weight (W) |
|---|---|
| $\alpha_1$ | 20.80e-6 |
| $\alpha_2$ | -0.10e-6 |
| $\alpha_3$ | 140.80e-6 |
| $\alpha_4$ | 190.80e-6 |
| $\alpha_5$ | 1.30e-6 |
| $P_{static}$ | 1.10e-3 |

Typically, the power consumption of a device is described as a per-access energy or a per-mode power, which only denote the energy and power consumption for a particular operating frequency; but these quantities are no longer valid if we vary the clock frequency. However, an *energy state machine* separates dynamic energy and static power, and is therefore valid at any clock frequency [10]. Figure 3 shows an example of a simple energy state machine. Each transition (arc) is annotated with its dynamic energy, and each state with a static power.

While an energy state machine is superior to the use of per-access energy and per-mode power, it is hard to characterize the dynamic energy and static power of a device using conventional power measurement methods. But cycle-accurate energy measurement [2] can distinguish between dynamic energy and static power.

## 3.2 PMU-based power estimation

### 3.2.1 Microprocessor

If a CPU comes with a PMU, there is no reason not to use it. Therefore we estimate the CPU core power consumption using the CPU PMU. We also use the relevant CPU PMU counters, such as $C[IEX]$ (instructions executed), $C[DDP]$ (data dependencies), $C[ICM]$ (instruction cache misses), $C[ITM]$ (instruction TLB misses) and $C[DTM]$ (data TLB misses). We use a linear power model [3] which assumes the following linear relations between the counter values and power consumption:

$$P_{CPU} = \alpha_1 C[IFM] + \alpha_2 C[DDP] + \alpha_3 C[DTM] + \alpha_4 C[ITM] + \alpha_5 C[IEX] + P_{static}, \quad (1)$$

where $\alpha_1, \cdots, \alpha_5$ are power coefficients, and $P_{static}$ is the static power consumption of the processor. The power estimation software periodically reads, resets and restarts the PMU counters, so that $P_{CPU}$ is the average power consumption of the CPU for a period $T_{pmu}$. The reference power measurement platform for verification is equipped with a TI OMAP5912 based on the ARM9, which does not have a CPU PMU. We therefore implemented a CPU PMU for the ARM9 core in our in-house system-level simulator. The target environments will be shown in detail in Section 4.

### 3.2.2 DDR SDRAM

Various types of memory transactions are supported by the DDR SDRAM, but we can significantly simplify the state transition diagram if we abstract its interaction with a cache memory as shown in Figure 4. The cache miss transactions are burst-mode transfers and their extent depends on the cache line size. State diagram in Figure 4 is simpler than the actual implementation, which is determined by the DDR SDRAM controller IP0.

The primary variation in the energy used by the DDR SDRAM that appears in Figure 4 is caused by i) auto-refresh operations (S17), ii) row misses and the resulting state transitions, which consist of precharging and row address (S3 $\rightarrow$ S0 $\rightarrow$ S1) when active-page mode is used, and iii) read or write operations (either S11 to

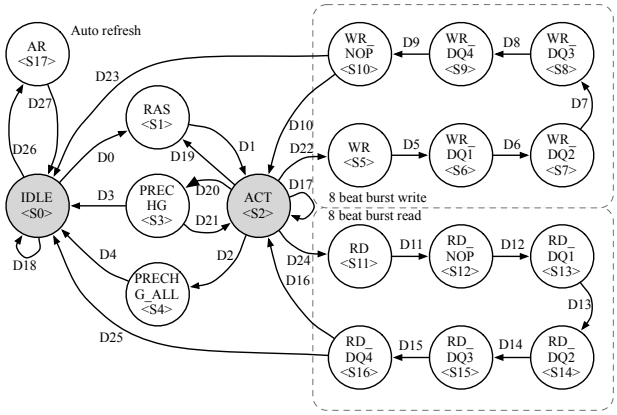


**Figure 4: The power and energy state diagram of the DDR SDRAM device.**

**Table 3: The static power and dynamic energy consumption of the a Samsung Mobile DDR SDRAM K4X51163PC devices (power: mW, energy: nJ).**

| State | Static power | Mode | Static power |
|---|---|---|---|
| S0 | 5.03 | S1 | 23.93 |
| S2 | 12.66 | S3 | 26.45 |
| S4 | 7.62 | S5, S6, S7, S8, S9, S10 | 17.02 |
| S17 | 2.61 | S11, S12, S13, S14, S15, S16 | 42.30 |

| Transition | Dynamic energy |
|---|---|
| D3, D4, D10, D16, D18, D23, D25 | 0.077 |
| D0 | 6.30 |
| D22+D5+D6+D7+D8+D9 | 5.83 |
| D24+D11+D12+D13+D14+D15 | 13.46 |
| D26 | 20.84 |
| D2 | 3.14 |
| D20 | 1.08 |
| D1, D17, D21 | 0.18 |
| D19 | 7.27 |

S16, or S5 to S10). When auto-precharge mode is used, a transition from precharge to idle (S4 $\rightarrow$ S0) occurs after every burst mode transfer. The proposed state-based PMU exactly tracks the DDR SDRAM internal state transitions, and profiles the correct number of state transitions as well as the number of cycles during which the SDRAM remains in each state. The DDR SDRAM PMU shown in Figure 4 has 18 state counters and 27 transition counters.

Table 3 shows the energy consumption coefficients for the Samsung Mobile DDR SDRAM K4X51163PC considered in this paper. All the energy values have been characterized by the cycle-accurate energy measurement setup described in Section 3.1. These coefficients are incoperated into a power estimation software, which simply reads the DDR SDRAM performance counters and multiplies the counts by the appropriate coefficients as follows:

$$P(DDR)(W) = \left( \sum_{i=0}^{26} Di \cdot C[DDR, T_i] \times 10^{-9} + \sum_{j=0}^{17} \frac{Sj \cdot C[DDR, S_j]}{f_{mem}} \times 10^{-6} \right) \frac{1}{T_{pmu}}, \quad (2)$$

where $f_{mem}$ is the memory clock frequency, $Di$ and $Si$ are coefficients from Table 3, and $C[DDR, T_i]$ and $C[DDR, S_j]$ are the $i$-th transition and the $j$-th state counters of the DDR SDRAM PMU.

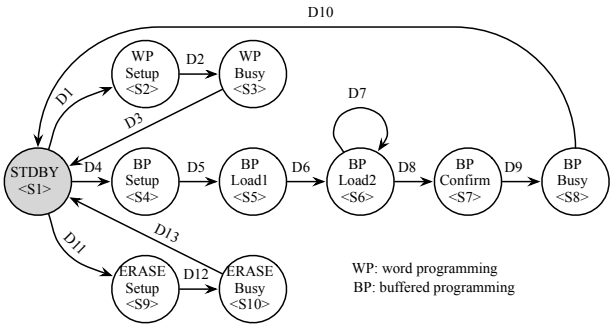


**Figure 5: The programming and erase state diagram of a NOR flash memory device.**

**Table 4: The static power and dynamic energy consumption of an Intel 28F256L18 NOR flash memory device (power: mW, energy: nJ).**

| Mode | Static power | Mode | Static power |
|---|---|---|---|
| S1 | 0.00 | S2, S4 ,S5, S6, S7, S9 | 6.30 |
| S3, S8 | 36.20 | S10 | 39.80 |

| Transition | Dynamic energy | Transition | Dynamic energy |
|---|---|---|---|
| D1 | 2.73 | D2 + D3 | 4.96 |
| D4 | 2.18 | D5 | 1.37 |
| D6 | 2.19 | D7 | 2.19 |
| D8 | 2.19 | D9 + D10 | 28.33 |
| D11 | 4.26 | D12 + D13 | 9.45 |

### *3.2.3 NOR flash*

NOR flash memory has four primary types of transaction: read, word programming, buffered programming and erase. Read requests initiate an SRAM-like read operation, which takes little power and is fast, although read requests also include the status register reads, which are also associated with the other three types of transaction. Write transactions transfer data from a CPU to a NOR flash for word and buffered programming. However, write transactions also include status register modifications that initiate erase and programming operations. Consequently, misinterpretation of NOR flash read and write transactions can result in huge errors in power estimation. Table 4 summarizes the static and dynamic energy consumption of the Intel 28F256L18 NOR flash memory. To support the state diagram in Figure 5, the PMU needs 13 transition counters and 10 state counters. The energy used by a NOR flash memory can be expressed in a way similar to (2). Note that the internal logic of the NOR flash in Figure 5 is not synchronized with the memory bus clock, and thus $f_{mem}$ is not defined. The PMU state machine is clocked by the PMU clock and we replace $f_{mem}$ with $f_{pmu}$ for the NOR flash.

## 3.3 Gate count and power overhead of the on-chip bus PMU

Table 5 shows the gate count and power overhead for the on-chip bus PMU for DDR SDRAM, and NOR flash. The two dominant area and power consumers are the AMBA AHB slave interface, and the registers for the performance counters and local variables. The three state machines require around 2k gates, and their power

**Table 5: Dynamic and leakage power and gate counts of the PMU for DDR SDRAM, and NOR flash, synthesized with the Synopsys toolchain (process: 0.18 $\mu$m, number of counters: 128 (24 bits), dynamic energy: pJ, and static power: $\mu$W).**

| IP | State machines | Registers | AHB slave IF |
|---|---|---|---|
| Ports | 132 | 58 | 298 |
| Nets | 879 | 10730 | 106 |
| Cells | 534 | 10666 | 34 |
| Gate count | 2037 | 34160 | 2622 |
| Dynamic energy | 16.23 | 568.40 | 43.98 |
| Static power | 0.33 | 6.5 | 0.55 |

**Table 6: Specification of the system-level energy simulator.**

| Component | Feature | |
|---|---|---|
| Simulator kernel | • SystemC 2.2.0 | |
| CPU core | • CPU core of SimIt-ARM 3.0<br>• ARM9TDMI, and XScale compatible | |
| Cache memory | • Level-1 I-cache<br>• Size: 16 KB<br>• 4-way associative<br>• Line size: 8 words | • Level-1 D-cache<br>• Size: 8 KB<br>• 4-way associative<br>• Line size: 8 words |
| System bus | • AMBA advanced high-performance bus (AHB)<br>• AHB cycle-level interface (AHBCLI) [11] compatible<br>• 32-bit address bus and 32-bit data bus<br>• Operating frequency: 96 MHz | |
| DDR SDRAM | • Samsung Mobile DDR SDRAM K4X51163PC<br>• Size: 512 Mb<br>• Bus width: 16 bits<br>• Operating frequency: 96 MHz (CL=3) | |
| NOR flash | • Intel 28F256L18 flash memory<br>• Size: 256 Mb<br>• Bus width: 16 bits<br>• Operating frequency: 48 MHz | |
| DMA controller | • Supports INCR16 burst transactions<br>• Internal FIFO size: 16 words | |

consumption is small. Adding more memory components requires additional gates for the registers and state machines, in proportion to the number of states and transitions. The gate count and estimated power overhead is not significant in comparison with these of an entire SoC.

# 4. PERFORMANCE EVALUATION

## 4.1 Experimental setup

### *4.1.1 Verification with a cycle-accurate simulator*

We developed a system-level energy simulator using the transaction-level modeling (TLM) facilities of SystemC. The components of the simulator include an instruction set simulator (ISS), a level-1 cache, an AMBA advanced high-performance bus (AHB), and several other AMBA-compatible peripheral device modules. All the modules of the simulator are coded in SystemC. The simulator was based on a SimIt-ARM ISS [12], which was modified to exhibit almost exactly the same instruction cycle times. The core ISS is connected to the cache module of the main simulator kernel by SystemC signals. The detailed specifications of each module are summarized in Table 6.

The simulator and the PMU share the same memory energy model. However, whereas the simulator sees every signal transition inside the memory devices, the PMU estimates the memory state changes from the AMBA AHB bus monitoring: this corresponds to the real situation. Due to the limited extent to which the status of memory devices can be observed, the PMU's estimates of power consumption naturally differ from that of the simulator. But we can use the simulator to tune the PMU, so as to bring the estimates as close as possible.

**Table 7: Accuracy of estimating the power consumption of a microprocessor using a CPU PMU (mW).**

| Method | Benchmark | | |
|---|---|---|---|
| | Basic math | MPEG4 decoder | MP3 encoder |
| Measured | 117.09 | 118.93 | 121.10 |
| CPU PMU | 118.75 | 120.20 | 118.73 |
| RMS error (%) | 1.6 | 2.47 | 5.21 |

We verified the accuracy of the power estimates of the proposed on-chip bus PMU using an in-house energy measurement platform based on a TI OMAP5912. The board architecture is quite typical, except for the special design of the power planes and memory system configuration. We divided the power plane of the printed circuit board into many power islands for each memory device so that we can measure their power consumption separately. This board is designed to preserve signal integrity and to minimize side effects. The memory system includes SRAM, SDRAM, mobile DDR SDRAM, NOR flash, NAND flash, and OneNAND flash. This over-configuration is only done so that a wide range of measurements can be made.

We hooked up a National Instrument data acquisition system to each power island, and measured the power consumption over time, while executing real benchmark programs. We then compared the estimate made by our on-chip bus PMU with the result from the full measurement system.

## 4.2 Experimental results

### *4.2.1 CPU power estimation*

First, we compared the estimated CPU(ARM926) power with the measured value. As in Section 3.2.1, we used a method of CPU power estimation comparable to previous work [3] to make a fair comparison. Table 7 shows around 2% to 5% RMS error between the estimates and the measurements for three different benchmark programs, which is quite satisfactory.

We compared the long-term average accuracy of power estimates made by a CPU PMU and by the proposed on-chip bus PMU. Table 8 summarizes the results for three different benchmark programs. As the CPU PMU is not capable of distinguishing SDRAM power from NOR flash power, it can only report the total memory system power. But, the on-chip bus PMU separately considers the SDRAM and NOR flash power, and forms estimates which are within 1% to 4% of the measured values.

Figure 6 shows the power consumption waveforms for an MP3 encoder, measured by a DAQ, and estimated by our on-chip bus PMU, and by a CPU PMU. Figure 6(c) shows that the CPU PMU is inaccurate because it misses most of the power peaks caused by memory accesses.

### *4.2.2 Comparison between a CPU PMU and the proposed on-chip bus PMU*

Next, we assessed the accuracy with which the proposed PMU estimates dynamic power consumption. Because a DAQ is unable to capture changes occurring over a few microseconds, we compared the PMU's results with those of the cycle-accurate simulator. Table 9 compares the RMS (root-mean square) errors between the CPU PMU and the on-chip bus PMU, and shows the superior accuracy of the on-chip bus PMU.

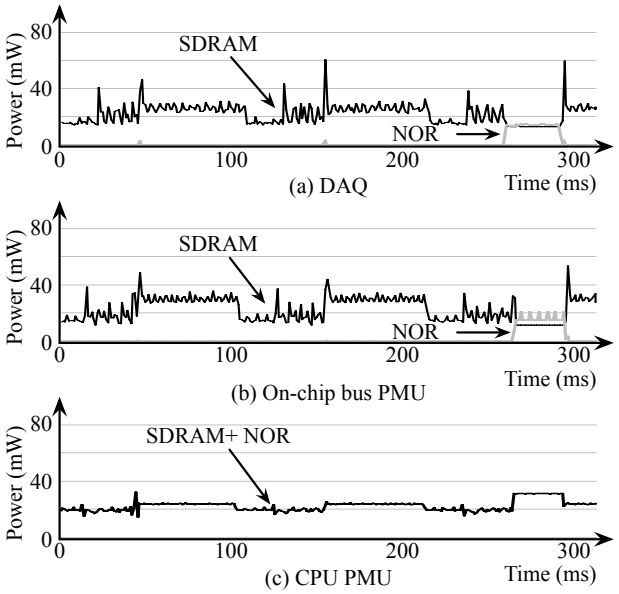


**Figure 6: Long-term average power comparison between a CPU PMU and the proposed on-chip bus PMU (MP3 encoder).**

**Table 8: Long-term average power estimation accuracy (mW).**

| Benchmark | Method | Memory | | | Error (%) |
|---|---|---|---|---|---|
| | | SDRAM | NOR | Sum | |
| *Basic math* | Measured | 13.31 | 0.02 | 13.33 | |
| | CPU PMU | N/A | N/A | 17.52 | 31.43 |
| | Proposed | 13.86 | 0 | 13.86 | 3.98 |
| *MPEG4 decoder* | Measured | 30.56 | 3.62 | 34.18 | |
| | CPU PMU | N/A | N/A | 28.27 | 17.29 |
| | Proposed | 31.54 | 3.80 | 35.34 | 3.39 |
| *MP3 encoder* | Measured | 22.92 | 0.66 | 23.58 | |
| | CPU PMU | N/A | N/A | 21.86 | 7.29 |
| | Proposed | 23.28 | 0.61 | 23.89 | 1.31 |

Figure 7 shows the differences between the power estimation waveforms for the CPU PMU and the on-chip bus PMU. Due to the limited extent to which the on-chip bus PMU can observe the memory state, there are still some missing peaks. Nevertheless, the results for the on-chip bus PMU are greatly superior to those for the CPU PMU. Note that the RMS error is somewhat exaggerated because it measures temporal mismatches, as well as mismatches of magnitude.

## 5. CONCLUSIONS

We have reported what we believe is the first attempt to achieve accurate system-level power estimation by means of an on-chip bus PMU, for a range of memory devices. While existing CPU PMUs estimate the energy consumption of memory by counting cache miss transactions, this new PMU traces the internal behavior of memory devices exactly to give a much better estimate of power consumption.

We have verified the proposed PMU using a system-level energy simulator and a real hardware prototyping board. Our PMU exhibits less than 5% average error compared with the real measurements. This is 6 to 10 times more accurate than conventional CPU PMU-based approaches. Automatic synthesis of a PMU to support DDR SDRAM, and NOR flash from an XML description requires 39K equivalent gate count.

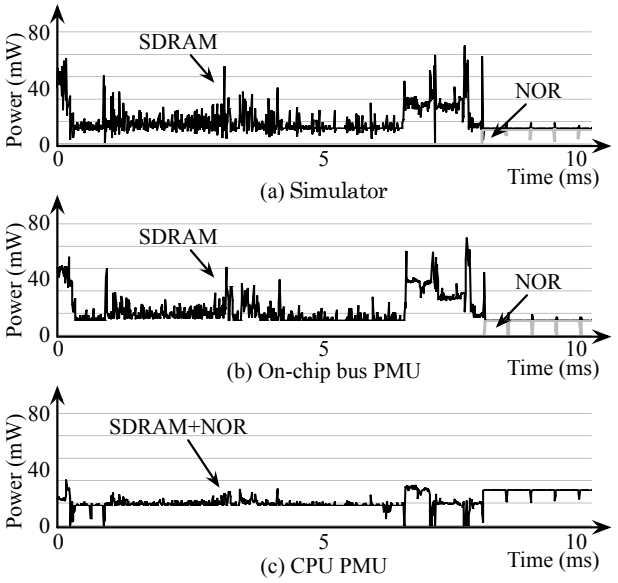


**Figure 7: Short-term dynamic power change comparison between a CPU PMU and the proposed on-chip bus PMU (MP3 encoder).**

**Table 9: Short-term power estimation RMS error (%).**

| Method | Benchmark | | |
|---|---|---|---|
| | Basic math | MPEG4 decoder | MP3 encoder |
| CPU PMU | 28.11 | 31.90 | 49.79 |
| Proposed | 2.63 | 17.91 | 18.14 |

## 6. REFERENCES


[1] J. Flinn and M. Satyanarayanan, "PowerScope: A tool for profiling the energy usage of mobile applications," in *WMCSA '99*, p. 2, 1999.

[2] N. Chang, K. Kim, and H. G. Lee, "Cycle-accurate energy consumption measurement and analysis: case study of ARM7TDMI," in *ISLPED '00*, pp. 185–190, 2000.

[3] G. Contreras and M. Martonosi, "Power prediction for Intel XScale® processors using performance monitoring unit events," in *ISLPED '05*, pp. 221–226, 2005.

[4] F. Bellosa, "The benefits of event: driven energy accounting in power-sensitive systems," in *SIGOPS EW '00*, pp. 37–42, 2000.

[5] I. Kadayif, T. Chinoda, M. Kandemir, N. Vijaykirsnan, M. J. Irwin, and A. Sivasubramaniam, "vEC: virtual energy counters," in *PASTE '01*, pp. 28–31, 2001.

[6] R. Joseph and M. Martonosi, "Run-time power estimation in high performance microprocessors," in *ISLPED '01*, pp. 135–140, 2001.

[7] I. Choi, H. Shim, and N. Chang, "Low-power color TFT LCD display for hand-held embedded systems," in *ISLPED '02*, pp. 112–117, 2002.

[8] W. L. Bircher, M. Valluri, J. Law, and L. K. John, "Runtime identification of microprocessor energy saving opportunities," in *ISLPED '05*, pp. 275–280, 2005.

[9] S. Gurun and C. Krintz, "A run-time, feedback-based energy estimation model for embedded devices," in *CODES+ISSS '06*, pp. 28–33, 2006.

[10] H. Shim, Y. Joo, Y. Choi, H. G. Lee, and N. Chang, "Low-energy off-chip sdram memory systems for embedded applications," *IEEE TECS*, vol. 2, no. 1, pp. 98–130, 2003.

[11] *AMBA AHB Cycle Level Interface (AHB CLI) Specification*, 2003. http://www.arm.com/products/solutions/ahbcli.html.

[12] W. Qin, "Simit-ARM." http://simit-arm.sourceforge.net/.