Validation of an Architectural Level Power Analysis Technique

Rita Yu Chen Robert M. Owens Mary Jane Irwin
Raminder S. Bajwa†
Department of Computer Science and Engineering
The Pennsylvania State University
†Semiconductor Research Laboratory
Hitachi America Ltd.

Abstract

This paper presents a technique used to do power analysis of a real processor at the architectural level. The target processor integrates a 16-bit DSP and a 32-bit RISC on a single chip. Our power estimator provides power consumption data of the architecture based on the instruction/data flow stream. We demonstrate the accuracy of the estimator by comparing the power values it produces against measurements made by a gate level power simulator for the same benchmark set. Our estimation approach has been shown to provide very efficient, accurate power analysis at the architectural level.

1 Introduction

As power dissipation has become a critical issue in many VLSI systems, power analysis at the architectural level has become more important because of its efficiency. It also allows the designer to experiment with design tradeoffs to lower power consumption. Most of the research in this area falls in the category of *empirical methods* which "measure" the power consumption of existing implementations and produce models based on those measurements. This *macromodeling* technique can be subdivided into three subcategories.

The first approach introduced in [7] is a *fixed-activity* macromodeling strategy called the Power Factor Approximation (PFA) method. The energy models are parameterized in terms of complexity parameters and a PFA proportionality constant. Thus, the intrinsic internal activity is captured through this PFA constant. This approach implicitly assumes that the inputs do not affect the switching activity of the hardware block.

To remedy this weakness of the fixed-activity approach, *activity-sensitive* empirical energy models have been developed. They are based on predictable input signal statistics, such as used in the SPA method [3, 4, 5]. Although the individual models built in this way are relatively accurate (the error rate is $10\% - 15\%$), overall accuracy may be sacrificed for the reasons of unavailable correct input statistics or an inability to model the interactions correctly.

The third empirical approach, *transition-sensitive* energy models, is based on input transitions rather than input statistics. The method presented in [2] assumes an energy model is provided for each functional unit - a table containing the power consumed for each input transition. The authors give a scheme for collapsing closely related input transition vectors and energy patterns into clusters, thereby reducing the size of the table for each functional unit. A significant reduction in the number of clusters, and also the size of the tables and the effort necessary to generate them, can be obtained while keeping the maximum error within 30% and the root mean square error within $10\% - 15\%$. After the energy models are built, it is not necessary to use any knowledge of the unit's functionality or to have prior knowledge of any input statistics during the analysis. However, this approach has never been used to analyze real commercial processors. Thus, it remains to be validated as a viable technique. Our work follows the third approach. The accuracy of our power estimator will validate the correctness of this methodology.

Our simulator imitates the behavior of the processor in each clock cycle as it executes a set of benchmark programs. It also collects power consumption data on the functional units (ALU, MAC, etc.) exercised by the instruction and its data. The results of our power estimator are compared with the power consumption data provided by [6]. This comparison confirms the accuracy of our power estimation technique.

The rest of this paper consists of four sections. Section 2 overviews the processor architecture and describes the simulation strategy. Section 3 presents the power estimation approach. Section 4 covers the power benchmarks and discusses the validation results. Finally, section 5 draws the conclusion and suggests the related future work.

2 Architectural Simulation of the Processor

2.1 Processor Overview

The processor considered in this research integrates a 32-bit RISC processor and a 16-bit DSP on a single chip. Figure 1 shows its main block diagram. The processor core includes the X-memory, the Y-memory, the buses, the CPU engine and DSP engine. The CPU engine includes the instruction fetch/decode unit, the 32-bit ALU, the 16-bit Pointer Arithmetic Unit (PAU), the 32-bit Addition Unit (AU) for PC increment, and 16 general purpose 32-bit registers. As the ALU calculates the X-memory address, the PAU can calculate the Y-memory address. Additional registers in the

1-58113-049-x-98/0006/$3.50

CPU support hardware looping and modulo addressing. The DSP engine contains the MAC, the ALU, the shifter, and 8 registers (six 32 bits wide and two 40 bits wide).

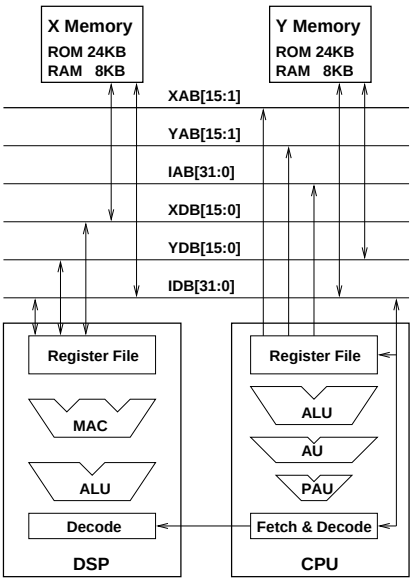

Figure 1: Block diagram of the processor

2.2 Architectural Simulation

An overview of the power estimator is shown in Figure 2. The inputs of the estimator are a benchmark program and energy models for each functional unit. The next section will introduce the details of energy model production. This section will only focus on the structural simulation of the architecture. The architectural simulator consists of several parts: the assembler, the controller, and the datapath implemented as many functional units (the small blocks labeled as m).

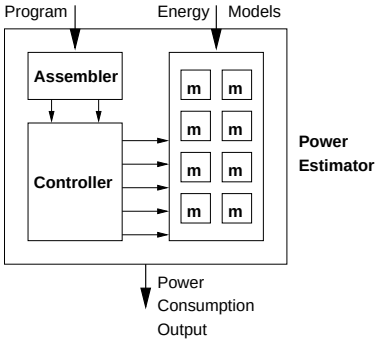

Figure 2: An overview of the power estimator

The first part of the simulation effort was to build an assembler which generates machine codes from benchmark programs written in the assembly language of the processor. The second part is the controller generating control signals for each functional unit when instructions are running on the simulator. These signals will be used to control the behavior of the functional units. For example, when a load instruction puts data from the memory to the IDB bus, the "MEMtoIDB" control bit will be set. The third part of the simulator specifies each functional unit in the datapath. As the control signals are set, the functional units are activated. For each unit, there is a function routine which gathers all event activities. Figure 3 shows an example of these specification functions. In this example, when a control signal is set, the IDB will receive data from one of its sources. This input data to the functional unit will be used to determine its power consumption.

```
toIDB()
{
  if ctr_bit[MEMtoIDB] is set
    read data from memory;
    put the data on IDB;
  if ctr_bit[MWBtoIDB] is set
    MWB buffer puts data on IDB;
  if ctr_bit[BIFtoIDB] is set
    BIF (Bus Interface) puts data on IDB;
}
```

Figure 3: Event activities in the IDB

Finally, the simulator will provide the precise behavior and functional unit state information. The behavior information describes which instruction is executed in each pipeline stage. The functional unit state information shows the content of each register or buffer, the data transferred on each active bus, and the function to be performed by each arithmetic unit (ALU, PAU, AU, MAC, etc.).

During the design of the simulator, the implementation of the pipeline was complicated because of the processor's unusual architecture. And it is necessary to use two program counters and special PC increment units, because the instruction stream for the processor is a mixture of 16-bit and 32-bit instructions. The simulator also implemented instruction loop buffers in the CPU engine.

3 Power Estimation of the Processor

Because dynamic power consumption is strongly dependent on the input switching of the functional units, using only the average power definitely causes a loss of accuracy. Our analysis technique overcomes this loss of accuracy by taking data and instruction streams into account.

The processor is separated into many functional units. The energy consumed by each functional unit per access can be computed as follows [1]

$$E_m = \frac{1}{2} C_m V dd^2 \qquad (1)$$

where C_m is the switch capacitance per access of the functional unit and Vdd is the supply voltage. For each active functional unit, C_m is calculated from the energy model of the unit based on the previous and present input vectors. The functional units can be grouped into two classes: *bit-dependent functional units* and *bit-independent functional units*. They have different energy characterization methods. Here, the terms energy and power are used interchangeably.

In the bit-dependent functional units, the switching of one bit affects other bit slice's operations. Typical bit-dependent functional units are adders, multipliers, decoders, multiplexers, etc. Their energy characterization is based on a lookup table consisting of a full energy transition matrix

where the row address is the previous input vector, the column address is the present input vector, and the matrix value is the switch capacitance. All combinations of previous and present input vectors are contained in one table. A major problem is that the size of this table grows exponentially in the size of the inputs. A clustering algorithm solves this problem and the sub-problems associated with it by compressing similar energy patterns. The details of this algorithm can be found in [2]. The capacitance data in the energy characterization table is obtained from a switch level simulation of the functional unit. The accuracy of switch level analysis is not as good as circuit level simulation tools, but is much faster.

In the bit-independent functional units, the switching of one bit does not affect other bit slice's operations; for example, registers, logic operations in the ALU, memories, etc. The total energy consumption of the functional unit can be calculated by summing the energy dissipation of the individual bits. To model the energy dissipation of a bus, load capacitance of each bit is used as an independent bit characterization. Furthermore, each bit is assumed to have the same load capacitance. The product of the load capacitance and the number of bits is the C_m in equation (1).

Some energy models are not built from the complete functional unit but from smaller subcells of these units. For example, because a 4:1 multiplexer can be made from three identical 2:1 multiplexers, its energy dissipation can be calculated by using the energy model of a 2:1 multiplexer. This reduces $2^{6^2} = 4096$ table entries to $2^{3^2} = 64$ entries. After building the subcell energy model, energy routines will implement those operations needed to calculate the functional unit power dissipation from subcells.

Because we did not have access to the design of the control unit of the processor, average data is used to characterize its power consumption. Although the accuracy is not as good as if we were able to build the transition dependent energy model, this average data reduces simulation time since there is no table lookup involved.

4 Validation Results

4.1 The Power Benchmarks

A set of simple synthetic programs listed in Table 1 are used as the benchmarks to validate our power analyzer. These are the same benchmarks used to test the processor in [6]. Power consumption of the instruction loop buffer (ILB), memories, buses, ALU, the multiplier and other functional units are collected as the benchmarks are run. Data values used were those which maximized the switching in the datapath.

Program	*Use ILB*	*Main Loop Operation*
pow032	No	padd pmuls movx+ movy+
pow033	Yes	padd pmuls movx+ movy+
pow035a	Yes	padd pmuls movx movy
pow034	No	padd pmuls
pow035	Yes	padd pmuls
pow035b	Yes	padd
pow035d	No	nop
pow035c	Yes	nop

Table 1: The Power Benchmark

In Table 1, "padd pmuls movx+ movy+" contains four separate operations executed simultaneously. They are an addition, a multiplication, two loads and two address increments. "Padd pmuls movx movy" is the same as the previous instruction except it contains no address increments.

4.2 Validation Results

The power simulation results generated by our architectural level power analyzer are compared with the data gathered by [6]. Overall, the average error rate of power analysis by our simulator is 8.98%.

Figure 4 compares the estimated (by our simulator) and reported (by [6]) power dissipation of the processor core including the CPU engine and the DSP engine. The power consumption data of each program is normalized to that of the *pow034* benchmark. As you can see, for most of the benchmarks our simulator produces results very close to those reported data in [6].

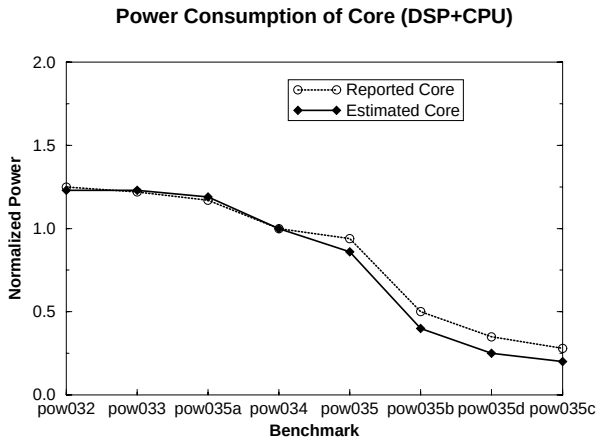

Figure 4: Core (DSP+CPU) estimated and reported power

In both the case of the *pow035d* benchmark and the *pow035c* benchmark, the power consumption is underestimated by our simulator. Possible reasons are that our power estimator has not considered clock power and that the power consumed by the control unit has not been accurately estimated. These two benchmarks use the least power overall, so that clock and control unit power account for a higher percentage of the total power consumption.

In order to verify the above suspicion, the simulation accuracy of each major functional unit (the DSP engine, the CPU engine, the memory and the top-level buses) was also analyzed and is shown in Figure 5. The estimated power distribution rate is compared to that reported by [6]. The distance from a symbol to the 1:1 line indicates the accuracy. The power simulation of the top-level buses is the least accurate of the four units - the average error rate is 10.2%. One of the reason for this is that the "top level interconnect" in [6] consists not only of the buses we considered but also the control and clock lines. The random control logic is also a large part of the total power consumption, but we were not able to build the energy model for the control unit since we did not have access to the design. Thus, the accuracy of the CPU engine power estimation is not as good as that of the DSP engine and memory, though the average error rate is only 5.7%.

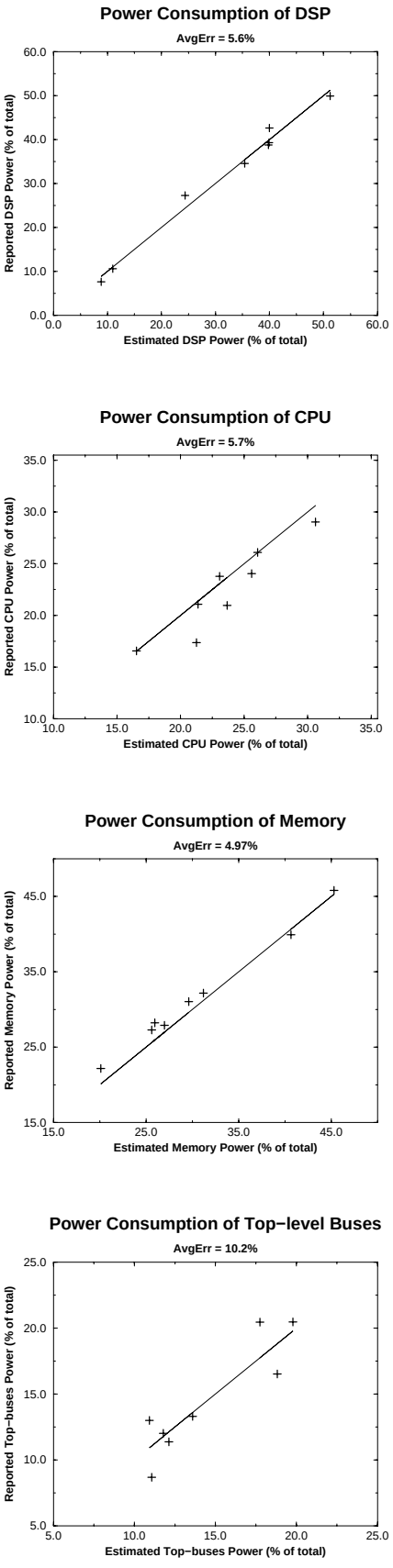

Figure 5: Power distribution comparison

5 Conclusion and Future Work

In this work, we implemented an accurate, high level power estimator for a real processor. Our estimation technique needs to do energy characterization for each functional unit (ALU, MAC, etc.) only once. When the operations of the functional units are specified by the instruction stream, the power consumed in each functional unit is calculated from its energy model. The simulation results clearly verified the correctness of our power analysis methodology. Without loss of accuracy, the running time of power estimation for each benchmark program was less than 90 CPU seconds. This time saving feature of our power estimator is beneficial for reducing the design cycle time. The structural simulator is also applicable for power optimization research at the architecture, system software, and application software levels.

The following are some of the factors which have not been taken into account during the power estimation:

- Clock power
- Transition dependent control logic power
- I/O pad power
- Varying signal arrival times
- The effects of functional unit placement and routing

Finding a better interconnect energy model also is a valuable research goal. Our future work to improve our power estimator will include all of the above. We are also investigating, running and testing more complicated benchmarks.

References

[1] H. Mehta, R.M. Owens, and M.J. Irwin. Instruction level power profiling. In *International Conference on Acoustics, Speech and Signal Processing* (1996).

[2] H. Mehta, R.M. Owens, and M.J. Irwin. Module energy characterization using clustering. In *Proceedings of 33rd Design Automation Conference* (June 1996).

[3] P. Landman, and J. Rabaey. Power estimation for high level synthesis. In *EDAC-EUROASIC* (1993), pp. 361–366.

[4] P. Landman, and J. Rabaey. Black-box capacitance models for architectural power analysis. In *1994 International Symposium on Low Power Electronics and Design* (April 1994), pp. 165–170.

[5] P. Landman, and J. Rabaey. Activity-sensitive architectural power analysis for the control path. In *1995 International Symposium on Low Power Electronics and Design* (April 1995), pp. 93–98.

[6] R. Bajwa, N. Schumann, and H. Kojima. Power analysis of a 32-bit RISC microcontroller integrated with a 16-bit DSP. In *1997 International Symposium on Low Power Electronics and Design* (1997), pp. 137–142.

[7] S.R. Powell, and P.M. Chau. Estimating power dissipation of VLSI signal processing chips: The PFA technique. In *VLSI Signal Processing IV* (1990), pp. 250–259.