Fast, Accurate Power Prediction for System-on-a-Chip (SoC) Processors

ABSTRACT

We describe a fast, accurate power prediction methodology for follow-on designs in the same System-on-a-Chip (SoC) architecture family. Our methodology cuts down power estimation times in design-cycles by enabling fast, accurate power estimates for individual components of the architecture. We demonstrate that we can achieve up to 99.97% correlation between simulation-based power estimates and silicon-based power measurements and simultaneously reduce overall design-cycle times. We use current generation silicon power measurements to calibrate pre-silicon simulation models for follow-on designs in the same fabrication process. We predict the power consumption on the silicon of a digital signal processor (DSP) called the Intel® Micro Signal Architecture (MSA) core on the Intel® PXA800F SoC for increased instruction widths and multi-cycle instructions within 3% average absolute error of silicon measurements.

1. INTRODUCTION

Embedded systems in battery-operated devices such as cellphones are becoming pervasive, creating a highly competitive and lucrative market. System-on-a-Chip (SoC) processors are increasingly being used in embedded systems to satisfy the growing needs of functionality and integration, allowing devices to become smaller and more powerful. SoC processor designs are often done in architecture families, with each new design improving upon the previous for different products or markets. The dual core architecture where a reduced instruction set computing (RISC) core handles general purpose applications and a digital signal processing (DSP) core handles data processing applications is the state-of-the-art design of modern cellular processors [8, 12]. Highly integrated SoCs that perform sophisticated functions consume proportionately higher power. This increase in the power consumption of the SoC reduces battery-life and causes thermal effects that can pose a significant threat to the life of the SoC and also the embedded product. The twin constraints of battery-life and heat dissipation have caused power to become an increasingly crucial design parameter in embedded systems.

In this paper, we describe a high speed and high accuracy methodology to predict silicon power in the follow-on designs of the current generation architecture in the same fabrication process. Our methodology cuts down power estimation time of design-cycles by accurately estimating component power using fine-grained switching activity control and a hierarchical differencing technique. Fine grained switching activity control refers to the user control over the switching activity of various microarchitectural components. Using fine grained switching activity control, we vary the switching activities of particular microarchitectural components through carefully hand-crafted programs, while avoiding the microarchitectural effects (cache miss latencies, branching overheads), the operating system effects (background processes, interrupts) and system effects (other cores and peripherals, if any on the SoC) from influencing the power estimates. Hierarchical differencing refers to a method of taking successive differences of power estimations to obtain component level power estimates at a finer granularity. Fine grained switching activity control combined with hierarchical differencing enables us to do faster component level power estimation. The hierarchical differencing technique not only helps to isolate the power dissipated by the components but also increases correlation accuracy of the simulation measurements as compared to the silicon measurements.

We use a single-instruction multiple-data (SIMD) statically scheduled DSP core called the Intel® Micro Signal Architecture (Intel® MSA) [4] on the Intel® PXA800F processor SoC [5, 8] to demonstrate and validate our methodology.

The rest of the paper is organized as follows. Section 2 introduces the related previous work. Section 3 gives the details about the Intel® MSA core microarchitecture. Section 4 explains the methodology. Section 5 presents the results and analysis of our methodology. Section 6 summarizes our contribution and suggests possible future directions.

2. PREVIOUS WORK

Power estimation can be classified as post-fabrication estimation and pre-fabrication estimation. Post-fabrication power estimation methodologies require processor silicon to estimate power. Tiwari et al. [10] assign the average measured power for each instruction to be the base power cost of that instruction and additional power of each pair of instructions as the overhead power cost for that pair of instructions. Sinha et al. [9] and Osqui [6] run each instruction in endless loops and thus build instruction level power models. The disadvantage of instruction level power analysis is that it fails to provide microarchitectural insights. For example, if no operands are switched, the base power cost of each instruction might simply be the clock circuitry power consumption. Chang et al. [3] build cycle-accurate energy models of various pipeline stages of a scalar RISC processor core by executing assembly programs on the processor evaluation board. Shafi et al. [7] use the IBM PowerPC® GP405 embedded SoC processor based evaluation board called Pecan, and run assembly experiments on the board with the involvement of an operating system to isolate the power consumed by various components. Post-fabrication techniques come later in the design cycle and provide little assistance in reducing the design times of the follow-on designs of the current generation architectures as these techniques require the silicon to be available.

Pre-fabrication power analysis methodologies help in getting early feedback on power characteristics of the design.

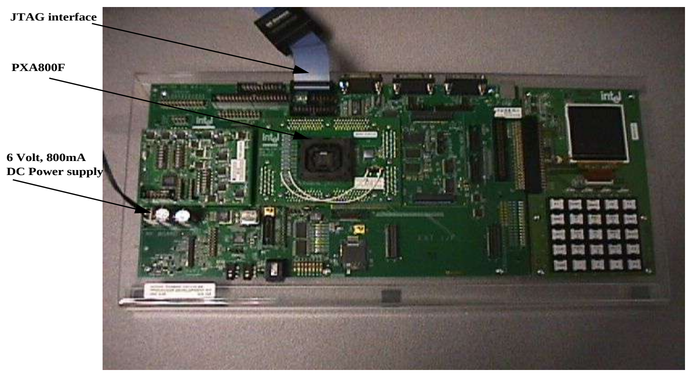
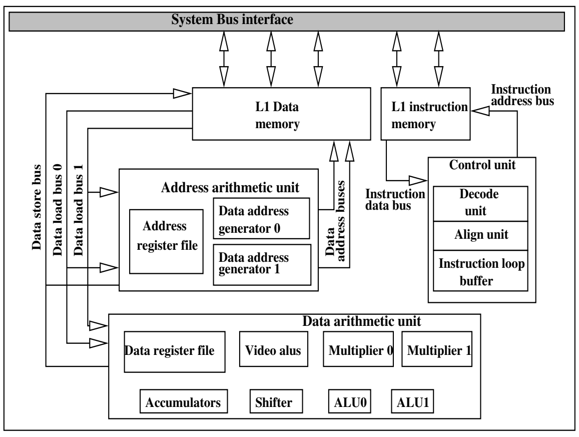


Figure 1: The Intel® PXA800F Processor Evaluation board



Source: R. Kolagotla et. al. "High Performance Dual–MAC DSP Architecture," (IEEE SPM 2002)

Figure 2: The Intel® MSA Architecture

They are simulation-based and do not require silicon. Several research efforts [1, 2, 11] have built detailed power models for various architectural components and integrated the power models into cycle-accurate architectural simulators. These power modeling techniques are tied to the particular logic structure of the microarchitectural components and are not readily usable for the case where the logic of the microarchitectural components is varied. Dynamic (switching) power is currently the largest component of power consumption. Switching activity based power analysis increases the accuracy of power estimations. However, switching activity based simulations are very time-consuming and resource intensive. Since the set of all the input vectors grows exponentially, simulating all possible input vector combinations is prohibitive. Simulation-based techniques might trade-off accuracy for speed thus resulting in conservative power estimates. If the designers rely on such conservative estimates to set the power/area budgets, they might miss the opportunity to aggressively optimize the designs. We need a way to prune the set of all possible inputs (input vector space) while obtaining power estimates that correlate well with silicon.

Our methodology uses the correlation between simulation power estimates and silicon power measurements to reduce the input vector space while simultaneously providing accurate power estimates. Since our methodology provides accurate power estimates, it provides opportunities for aggressive optimizations to logic and circuit designers. Our methodology can reduce the design cycle times in the current generation architecture through component level power estimation. While differencing is used in previous work to estimate the power of architectural components, we show that differencing also increases correlation accuracy between the simulation-based power estimates and the measured silicon power.

3. EXPERIMENTAL SET-UP

The experimental set-up (Figure 1) consists of the Intel® PXA800F processor SoC [8] based evaluation board (EVB). The Intel® PXA800F SoC processor has integrated the Intel XScale® core [13], the Intel® MSA core, the clock/power controller and peripherals in a 130 nanometer low leakage flash + logic process technology. The EVB communicates with a host computer through the JTAG interface shown in Figure 1. We use the JTAG interface to transfer the executable program from the host machine to the EVB. Note that the EVB does not need an operating system for program execution, thus eliminating the effect of operating system processes on the power measurements.

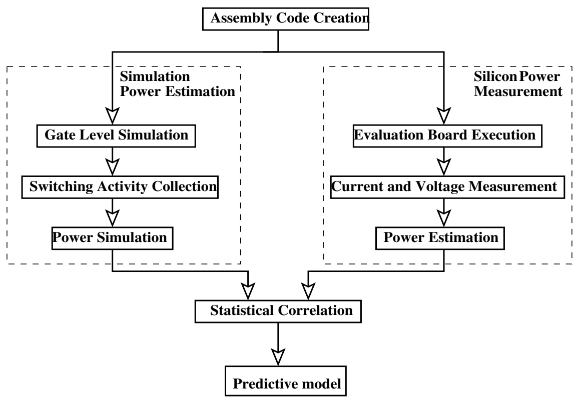
3.1 The Intel® MSA Core Micro-architecture

We describe the high-level microarchitectural features of the Intel® MSA core as shown in Figure 2. The Intel® MSA core is a SIMD DSP core with variable instruction widths of 16-bits, 32-bits and 64-bits. It has a modified Harvard architecture with eight fully interlocked pipeline stages. Instructions and data reside in separate level one (L1) memories but share a common level two (L2) memory. The core consists of a control unit, an address arithmetic unit, and a data arithmetic unit. The control unit consists of a decode unit, an align unit, and instruction loop buffer (ILB) circuitry. The address arithmetic unit consists of an address register file and two data address generators. The address register file includes an 8-entry pointer register file (registers p0-p7) and a 4-entry index register file (registers i0-i3) used for memory addressing purposes. The data address generators generate the memory addresses needed to access the data memory. The data arithmetic unit consists of an 8-entry 32-bit data register file, four 8-bit video ALUs, two 16x16-bit multipliers, two 40-bit accumulators, one 40-bit barrel shifter and two 40-bit ALUs.

We use an existing architectural feature of the Intel® MSA core called the instruction loop buffer (ILB) in our methodology to minimize the effects of instruction memory hierarchy and branching, thus increasing the accuracy of power estimates. The ILB is a 64-bit wide, 4-entry deep buffer that enables the execution of instruction loops with four or less instructions without any branching overhead.

4. METHODOLOGY

In this section, we describe the methodology shown in Figure 3. The goal of the methodology is to devise a model to predict the power consumption of follow-on designs of the current architecture. We estimate the power consumption using simulation for a set of tests and run the same set of tests on the evaluation board and measure the power as



Assembly Code Creation
Simulation Power Estimation
Silicon Power Measurement
Gate Level Simulation
Switching Activity Collection
Power Simulation
Evaluation Board Execution
Current and Voltage Measurement
Power Estimation
Statistical Correlation
Predictive model

Figure 3: Power Prediction Methodology

shown in Figure 3. We then plot the simulation and silicon power measurements to observe statistical correlation. We numerically fit a predictive model to predict the power consumption of follow-on designs of the current architecture.

We start by estimating the power of various microarchitectural components as shown in Figure 4. First, we select a microarchitectural component as the focus of the experiment. Then, we create three assembly programs, one that does not switch the microarchitectural component under consideration (base experiment), another that switches the microarchitectural component to the minimum extent we could devise (minimal switching experiment) and another that switches the microarchitectural component to the maximum extent (maximal switching experiment). We carefully vary the switching activity of the component in relation to the rest of the system, while ensuring that the rest of the system does not excessively influence the activity of the component. We use the ILB to avoid the microarchitectural effects of instruction memory hierarchy and branching on power estimates. We configure the L1 instruction memory and the L1 data memory as SRAMs and not as caches to avoid cache miss latencies from influencing the power estimates. We do not involve an operating system in our methodology thus eliminating operating system effects (background processes and interrupts) on the power estimates. We configure the Intel XScale® core to be in an inactive state and disable all peripherals to avoid system level effects on power estimates.

The power estimation equations for the base program, minimal switching program and maximal switching program are shown in equation (1), equation (2) and equation (3) respectively in terms of the datapath power P_{dp}, clock power P_{clock}, leakage power P_{leak} and control path power P_{cp}. Note that clock circuitry (P_{clock}), leakage (P_{leak}), and control path (P_{cp}) consume identical amounts of power in both minimal switching and maximal switching tests and cancel out by the differencing operation.

$$
\begin{aligned}
P_{base} &= P_{dp_{nop}} + (P_{clock} + P_{leak} + P_{cp}) && (1)\\
P_{tot_{min}} &= P_{dp_{min}} + (P_{clock} + P_{leak} + P_{cp}) && (2)\\
P_{tot_{max}} &= P_{dp_{max}} + (P_{clock} + P_{leak} + P_{cp}) && (3)\\
P_{min} &= P_{tot_{min}} - P_{base} = P_{dp_{min}} - P_{dp_{nop}} && (4)\\
P_{max} &= P_{tot_{max}} - P_{base} = P_{dp_{max}} - P_{dp_{nop}} && (5)\\
P_{switch} &= P_{tot_{max}} - P_{tot_{min}} = P_{dp_{max}} - P_{dp_{min}} && (6)
\end{aligned}
$$

Equation (4) gives the power contribution of the microarchitectural component when the component has zero/negligible switching activity. Equation (5) gives the power contribution of the microarchitectural component when the component has very high switching activity. Equations (4) and (5) give the respective minimum and maximum contributions for component power dissipation and could be used by circuit designers to investigate and optimize circuits. By taking the differences of Equations (4) and (5), we can isolate the power of specific microarchitectural components. Equation (6) gives the power needed to switch the microarchitectural component from a state of negligible switching activity to a state of maximal switching activity. We call this technique of series of differencing operations, which enable us to obtain fine-grained power estimates, as hierarchical differencing.

instruction_loop_buffer_start:	instruction_loop_buffer_start:
nop;	p0 = [i0++];
nop;	p0 = [i0- -];
nop;	p0 = [i0++];
instruction_loop_buffer_end:	instruction_loop_buffer_end:
nop;	p0 = [i0- -];

Table 1: Programs to Cause the Load Circuitry to be (a) Unused (b) Used

As an example, we discuss how to isolate the power consumed by the data load circuitry of the Intel® MSA core. The data load circuitry includes the address arithmetic unit, the data address buses and the data load buses between the L1 data memory and the address arithmetic unit (see Figure 2). Figure 5 graphically depicts the active (continuous line) portion and inactive (dashed line) portions of the data load circuitry for each of the three cases corresponding to equations (1), (2) and (3).

We create an assembly program that does not use the data load circuitry by filling the instruction loop buffer (ILB) with "nop" instructions and executing out of the ILB. Table 1(a) shows the example program (base program) that does not switch the data load circuitry. We preload the L1 data memory with the required data values. In the base case, the data load circuitry is not being used since we are executing nops. We run the base assembly program and measure the power P_{base}.

We then create a modified assembly program that minimally switches the data load circuitry. Table 1(b) shows an example program that switches the data load circuitry. We preload the index register i0 with the address of the L1 data memory location being accessed. The pointer register p0 loads values from the two adjacent memory locations pointed to by the index register i0 with *post-increment* and *post-decrement* operations. We preload the two adjacent data memory locations with the same value e.g., 0x00000000 so that there is no switching on the data load bus. The data address generator of the address arithmetic unit performs post-increment and post-decrement operations to access the two adjacent memory locations. In the minimal switching case, the address arithmetic unit generates address references to the two adjacent data memory locations that have the same preloaded data value, thus resulting in zero activity on the data bus. We run the modified assembly program and calculate power, $P_{tot_{min}}$.

We then modify the data in the same two adjacent mem-

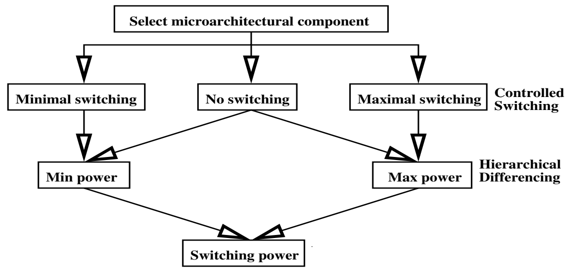


Figure 4: Power Estimation Procedure

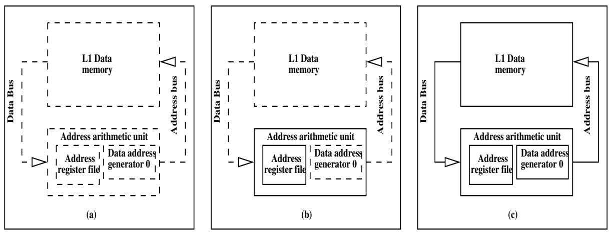


Figure 5: (a) Base (b) Minimal (c) Maximal Switching Cases of the Data Load Circuitry

ory locations to be bit-complementary (e.g., 0x00000000 and 0xFFFFFFFF) resulting in maximum switching on the data load bus of the data load circuitry. We run the modified assembly program, taking care to maintain the same experimental conditions and calculate power, $P_{tot_{max}}$. In the maximal switching case, the address arithmetic unit generates address references to two adjacent data memory locations that have preloaded bit-complementary values, thus resulting in maximum activity on the overall data load circuitry. We designate the difference between the base experiment power, P_{base}, and the modified experiment power, $P_{tot_{min}}$, as the power, P_{min}, of the data load circuitry with minimal switching. We designate the difference between the base experiment power, P_{base}, and the modified experiment power, $P_{tot_{max}}$, as the power, P_{max}, of the data load circuitry with maximal switching.

4.1 Simulation-based Power Estimation

Simulation-based power estimation consists of three steps (Figure 3): gate-level simulation, switching activity collection and power estimation. The simulator tool simulates the test program on the post-layout (annotated with RC parasitics) netlist of the Intel® MSA core and captures the switching activity. The switching activity is collected in a hierarchical format enabling us to isolate the switching activities of individual microarchitectural components. The power estimation tool uses the hardware netlist description of the component along with the corresponding switching activity of the component to estimate the power of the component.

Note that the simulation is performed only once to collect the switching activities of all microarchitectural components. After collecting the switching activity, we perform power analysis of individual components simultaneously by using the corresponding switching activities and netlists of the individual components, thus reducing the overall power estimation time. Since the input vectors fed to the different components belong to a single test program and collected in the same time window, the individual power estimates of the components can be summed to obtain the total power for that test program. In Section 5.2, we validate our methodology by demonstrating that simulation-based component power estimation tracks silicon power very closely.

Test name	Example
Addition using data registers	r3 = r1 + r2
Addition using pointer registers	p3 = p1 + p2
Logical and	r3 = r1 & r2
Logical or	r3 = r1 | r2
Pointer register to pointer register move	p2 = p1
Data register to data register move	r2 = r1
Logical xor	r3 = r1 ˆ r2

Table 2: A Sub-set of Tests Used for Correlation

4.2 Silicon-based Power Measurement

Silicon-based power measurement consists of three steps (Figure 3): running the test on the EVB, measuring the true root mean square (RMS) current and voltage for the duration of the program execution, and calculating power as the product of the measured current and the voltage. We run the same set of assembly programs used in simulation-based power estimation on the evaluation board (EVB) and measure the true RMS current I and true RMS voltage V. Since we do not involve any operating system in our methodology, operating system effects (background processes and interrupts) do not influence the measurements. Since we configure the Intel XScale® core to be inactive and disable all peripherals, they contribute a common bias to all the experiments and is removed by the differencing operation. We repeat the experiments multiple times in the same ambient conditions and calculate the average power to prevent variations in temperature-dependent leakage power of the SoC processors.

The power estimation equations for the base test, minimal switching test and the maximal switching test are identical to that shown in equations (1) through (6), except for the additional system power consumed by the Intel XScale® core in an inactive state and with peripherals disabled. In particular, the system power is the sum total of the power of I/O pads, leakage power on the Intel XScale® core and idle mode power of peripherals. System power is equally present in all the tests and cancels out by taking the differences among the equations (1) through (3.)

5. RESULTS

We chose a set of tests to be representative of the data processing operations most frequently performed by the DSP core. Table 2 shows a sample list of tests. In the first set of results, we demonstrate how fine grained switching activity control leads to more accurate correlations between simulation-based power estimates and silicon-based measurements. We also demonstrate that the differencing operation further increases the correlation accuracy between the two estimates.

5.1 Accurate Power Model

Figure 6 shows the comparison of power obtained from the silicon measurements versus pre-silicon simulation estimates for the maximal switching tests ($P_{tot_{max}}$) (no differencing).

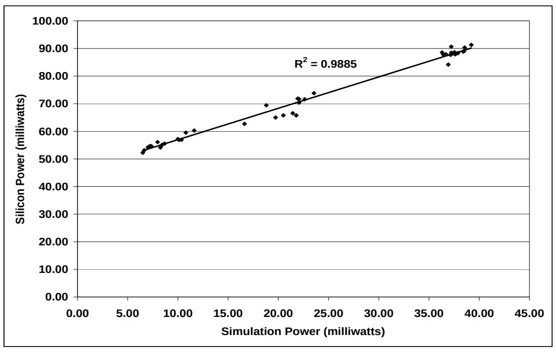


Figure 6: Power $P_{tot_{max}}$ Correlation for Data Processing Components (no differencing)

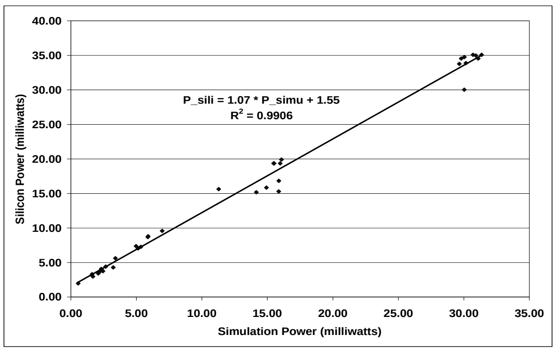


Figure 7: Power P_{switch} Correlation for Data Processing Components (differencing)

Component	Percent Correlation (%)
Data arithmetic unit	98.56
Address arithmetic unit	97.05
Register File	99.97

Table 3: Correlation of Components

For the case of maximal switching tests, simulation-based power tracks the silicon-based power with a correlation accuracy of 98.85%. In particular, note the system bias introduced by the inactive Intel XScale® core and peripherals in the silicon-based power measurement estimates. A similar comparison (Figure 7) of the differential power estimates (P_{switch}) obtained from simulation versus silicon showed a correlation of 99.06%. The common bias introduced by the two heterogeneous power estimation setups are removed by the differencing operation thus enabling us to accurately track the trends. In particular, differencing has removed the respective common bias in the simulation-based power and the silicon-based power. Since the trends track each other in a systematic manner, we can use this feature for design space exploration of future architectures without actually having the silicon as long as we use the same technology and the same base microarchitecture.

5.2 Power Estimation Time Reduction

In this section, we describe the reduction in power estimation times that can be obtained using our methodology. We carefully create tests to vary the switching of only one microarchitectural component. The rest of the components are either unused (clock-gated) or contribute equal switching to all test cases (common bias). Prior to the development of our methodology, the standard practice was to simulate the entire design. With our methodology, it suffices to simulate only the component of interest thus resulting in a huge savings of time and resources to run simulations. As an example, we configure the experiments in such a way as to make the data arithmetic unit the dominant power consuming unit. We correlate the simulation-based power of the data arithmetic unit with the silicon-based power measurements. The simulation power ($P_{tot_{max}}$) estimates versus the silicon power estimates show a correlation of 98.56%. To ensure the generality of this methodology, we repeat the same procedure for the entire datapath of the DSP core (about 60% coverage in terms of total transistor count) and observed a high/similar correlation for all of our experiments. Table 3 shows a list of components and the correlation obtained. In particular, note that the register file tests involve moving values from the general purpose registers in the data arithmetic unit to the pointer registers in the address arithmetic unit. Thus, the component power estimation can be extended to cases where multiple components are switching by summing the corresponding power contributions.

It is not as straightforward to design tests to control the switching activities of components such as the instruction cache or control logic, but the authors believe that it is feasible, especially after accurately estimating the power contribution of a majority of datapath components.

5.3 Power Prediction in Follow-On Designs

In this section, we demonstrate how we use the correlation between the simulation-based power estimates and silicon-based power measurements to predict the power for follow-on designs. The instruction widths and pipeline stages are important scaling parameters in architecture families. So, we investigate the utility of our methodology with respect to increasing instruction widths and multi-cycle instructions. We build a predictive power model using the 16-bit instructions that perform data processing operations in one cycle and use the predictive power model to predict the power consumed by 32-bit instructions that perform data processing operations and power consumed by multi-cycle instructions (that take more than one cycle for execution). For the 16-bit instructions, the silicon power (P_{sili}) is related to the simulation power (P_{simu}) according to Equation (7). Equation (7) is derived by curve-fitting the values plotted in Figure 7. The gradient of Equation (7) is 7% off from a perfect fit (gradient=1) owing to process parameter variations. Also, the constant 1.55 milliwatts accounts for inaccuracies in estimating the leakage power.

$$P_{sili} = 1.07 * P_{simu} + 1.55(mW) \qquad (7)$$

We obtain the simulation-based power estimates (P_{simu})

Test	Differential Power(mW) Silicon Predictive Model	Differential Power(mW) Silicon Measurement	% Error Absolute
absolute	37.03	38.83	4.62
non-saturating add	34.53	34.43	0.29
saturating add	33.21	34.87	4.75
maximum	35.52	35.20	0.91
minimum	35.67	36.52	2.32
non saturating subtract	34.12	33.66	1.36
saturating subtract	34.29	36.52	6.09
multiply-accumulate	17.11	17.27	0.94
		Average Absolute Error	2.66

Table 4: Prediction of Silicon Power Consumption for Wider and Multi-cycle Instructions

for the 32-bit instructions and multi-cycle instructions. Using Equation (7), we obtain the corresponding silicon-based predictive estimates (P_{sili}). We run the same set of experiments on the EVB and compare the predicted power with the actual silicon power in Table 4. Note that our model predicts the silicon power within an average absolute error of 2.66%. In particular, note that while the multiply-accumulate instruction is both a 32-bit instruction and also spans over two pipeline stages (2-cycle latency), our approach still provides a power estimate that is very close to the silicon power consumption.

6. SUMMARY

We demonstrated a methodology that obtains accurate correlation between simulation-based power estimates and silicon-based power measurements. We applied the methodology to reduce the power estimation times through component power estimation, thus enabling faster design space exploration. Logic design engineers and circuit design engineers can use our methodology to quickly identify the microarchitectural components consuming high amounts of power. Software architects and compiler writers can use the results of our methodology to improve the power-optimization heuristics used in instruction selection, instruction scheduling and register allocation. We demonstrated the utility of our methodology to follow-on designs of the current generation architectures by predicting the power accurately for increasing instruction widths and multi-cycle instructions.

An interesting extension to the methodology is to explore power-performance trade-offs and understand the energy characteristics of applications. Another interesting extension is to explore the applicability of the methodology in the design of general purpose processors.

7. REFERENCES

[1] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," *Proceedings of the 27th International Symposium on Computer Architecture*, pp. 83-94, Piscataway, NJ, 2000.

[2] G. Cai and C. Lim, "Architectural Level Power/Performance Optimization and Dynamic Power Estimation," *Cool Chips Tutorial: An Industrial Perspective on Low-Power Processor Design*, IEEE CS Press, pp. 90-113, Los Alamitos, CA, 1999.

[3] N. Chang, K. Kim and H. Lee, "Cycle-accurate energy consumption measurement and analysis: Case study of ARM7TDMI," *Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, pp. 185-190, 2000.

[4] R. Kolagotla, J. Fridman, B. Aldrich, M. Hoffman, W. Anderson, M. Allen, D. Witt, R. Dunton, L. Booth, "High performance dual-MAC DSP architecture," *the IEEE Signal Processing Magazine*, Vol. 19(4), pp. 42-53, 2002.

[5] D. Krishnaswamy, R. Stevens, R. Hasbun, J. Revilla and C. Hagan, "The Intel® PXA800F Wireless Internet-On-A-Chip Architecture and Design," *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 39-42, 2003.

[6] M. Osqui, "Evaluation of Software Energy Consumption on Microprocessors," *Masters Thesis, Massachusetts Institute of Technology*, 2001.

[7] H. Shafi, P. Bohrer, J. Phelan, C. Rusu, and J. Peterson, "Design and Validation of a Performance and Power Simulator for PowerPC Systems," *The IBM Journal of Research and Development*, Vol. 47(5/6), 2003.

[8] The Intel® PXA800F Cellular Processor, *http://www.intel.com/design/pca/prodbref/252336.htm*

[9] A. Sinha and A. Chandrakasan, "JouleTrack - A Web Based Tool for Software Energy Profiling," *Proceedings of the 38th Design Automation Conference*, pp. 220-225, 2001.

[10] V. Tiwari, S. Malik, and A. Wolfe, "Power Analysis of Embedded Software: A First Step Towards Software Power Minimization," *Proceedings of International Conference on Computer-Aided Design*, pp. 384-390, 1994.

[11] N. Vijaykrishnan et al., "Energy-Driven Integrated Hardware-Software Optimizations Using SimplePower," *Proceedings of the 27th International Symposium on Computer Architecture*, pp. 83-94, Piscataway, NJ, 2000.

[12] The Texas Instruments OMAP® Technology, *http://focus.ti.com/omap/docs/omapgenpage.tsp?navigation Id= 9508&templateId=5663&path= templatedata/cm/omapovw/data/gproc*

[13] The Intel® PXA250 Processor with XScale® Technology, *http://www.intel.com/design/pca/prodbref/298620.htm*