Fine-grained On-line Power Monitoring for Soft Microprocessor based System-on-Chip

Young H. Cho and Siddharth S. Bhargav
Information Sciences Institute
University of Southern California
Marina del Rey, California, U.S.A.
{youngcho, ssbharga}@isi.edu

Abstract — **Today's CMOS technologies allow larger circuit designs to fit on a single chip. However, this advantage comes at a high price of increased process-voltage-temperature (PVT) variations. FPGAs and their designs are no exceptions to such variations. In fact, the same bit file loaded into two different FPGAs of the same model can produce a significant difference in power and thermal characteristics due to variations that exist within the chip. Since it is increasingly difficult to control physical variations through manufacturing tasks, there is a need for practical ways to sense chip variations to provide a way for circuit designers to compensate or avoid its negative effects. One of the most critical aspects of such variation is power. Therefore, we developed and demonstrated a high accuracy on-chip on-line Energy-per-Component (EPC) measurement technology on Xilinx FPGAs since 2011. However, we found that the hardware overhead associated with such method limited the use of the technology. Therefore, our follow-up work in Energy-per-Operation (EPO) on Spartan FPGA with OpenRISC SoC produced an equally accurate power monitoring technology with drastically lower hardware overhead. While this method made our technology more practical for SoC designs on FPGAs, it did not produce component level power dissipation data that previous EPC method provided. Therefore, we extend this prior work with a new algorithm to extract EPC values from EPO result. Despite the lower hardware overhead, this change ended up improving the accuracy of the power result by unraveling the instruction-level abstraction into component-level energy consumption.**

Keywords—power monitor; power variation; integrated circuits; computer architecture; system-on-chip; soft microprocessor; FPGA; PVT;

I. INTRODUCTION

Newer CMOS technologies allow the larger design to be integrated on a single die. However, the power density of new chip is much higher and its process-voltage-temperature (PVT) variation characteristics are more difficult to predict and control. Due to higher complexity and lower predictability of physical behavior of new chips, model-based tools no longer play a significant role in improving chip performance or yield. Because of this, newer designs integrate some form of on-line power estimation, dynamic power manager, and adaptive task scheduler to compensate for the negative effects of worsening variations.

In computer system designs, power management methods have shown to improve computer efficiency [2]. However, one survey performed about two decades ago suggests that macro-level power management techniques may have reached their performance limits [6]. Therefore, current research efforts seem to point to fine-grained power management as the path forward to additional power savings for the future digital designs [1].

An enabling technology for an effective power management is high accuracy fine-grained power measurement system. Since traditional power measurement algorithms rely on statistically derived power models and direct measurement methods, they are proving to be either not accurate enough or too expensive to drive fine-grained power manager. FPGAs present additional challenges to measurement systems because of their need to support hardware reconfiguration.

We present an efficient on-line sub-component level power measurement method that is fine-grained and highly accurate. In the following section, we, first, present a case study that shows measurable variations that exist within the same chip as well as across different chips of the same FPGA model. Then we discuss the core concept of the technology in section 3. For the context, we summarize our earlier application of the concept on a microprocessor-based SoC in section 4. In section 5, we present our new conversion technique in extracting energy-per-component (EPC) from operation counts. Finally, we discuss and compare our results in the last sections of the paper.

II. PRELIMINARY TEST RESULT

To verify our hypothesis on the significance of variations within and amongst the same model of FPGAs, we conducted a power measurement case study.

For this study, we instantiated the OpenRISC SoC (ORPSoC) on Digilent Atlys board instrumented with a current sense resistor and a 1 kHz differential ADC (PMODADC5) [30]; we computed the power by directly measuring the load voltage and the voltage drop across the current sense resistor. The ORPSoC design was compiled four times to generate four heuristically produced different physical layouts for FPGA.

For our experiment, we repeated Dhrystone benchmark for OpenRISC processor 16 times while measuring FPGA power consumption. Table 1 presents the average energy deviation of 16 different combinations of bit files and FPGAs compared one against another.

For each FPGA, we observe power consumption variations between different bit files to range from 3.10% to 9.31% (FPGA 1-4). For the same bit file across different FPGAs, we observe variation that ranges from 5.08% to 10.91% (bit file 1-4). Furthermore, the table suggests a non-deterministic distribution of variation across on-chip devices since power

This material is based in part upon work supported by the NSF under Grant Numbers IIP-1414397, IIP-1622922, and IIP-1700844. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

978-1-5386-3797-5/17/$31.00 ©2017 IEEE

deviation from one bit file to another does not change in the same increasing or decreasing order based on which FPGA chip was programmed.

Such range of variations in measurement suggests that simulated power estimation software will not be able to produce a highly accurate result. In fact, to ensure guaranteed performance across all FPGAs, the simulator must produce the most conservative estimates.

	FPGA 1	FPGA 2	FPGA 3	FPGA 4
Bit file 1	0.00%	2.10%	3.30%	-1.78%
Bit file 2	2.50%	3.60%	-1.98%	1.82%
Bit file 3	4.80%	5.20%	2.09%	-2.19%
Bit file 4	6.10%	3.90%	-3.79%	7.12%

Table 1: Measured power difference between 4 different bit files of the same design on 4 different chips of the same FPGA model

III. GroundWork

Given non-deterministic variation on every single FPGA, we developed a minimally invasive online technology that accurately determines sub-component level power consumption [26-30]. The core concept and the groundwork leading up to the contributed research in this paper is reviewed in this section to establish the context.

$$P = \sum_{j=1}^{n} q_j a_j s_j + L + R \qquad (eq.\ 1)$$

In the power eq. 1, L component represents the lumped sum of leakage power and R represents residual component while $q_j a_j s_j$ (q = variation factor, a = activity parameter, $s = cv^2f$). Since each type of representation is an abstraction of circuit operation with a different level of detail, their results reflect a different level of accuracy and overhead.

For modern circuits, accounting for activity rate at every gate or even at the level of circuit cone incurs impractical hardware instrumentation overhead. Therefore, we show in [30] that our technology applied to microprocessor-based System-on-Chip (SoC) can be optimized to remove a large portion of the hardware sensors.

Therefore, we detect and account for issuing of each type of SoC operation (expressed as different combinations of opcodes and referenced addresses). Since each of these k-operation activates the same subset of gates fixed number of times, we can rewrite equation 1 to represent power consumed by k components.

If we let $w_j = q_j s_j$ then, eq.1 can be reformulated as:

$$P_{meas} = \sum_{j=1}^{m_1} a_j w_j + \sum_{j=m_1+1}^{m_2} a_j w_j + \cdots + \sum_{j=m_{k-1}+1}^{n} a_j w_j + L + R$$

$$P_{meas} = a'_1 w'_1 + a'_2 w'_2 + \cdots + a'_k w'_k + L + R$$

$$P_{meas} = \sum_{j=1}^{k} a'_j w'_j + L + R \qquad (eq.\ 2)$$

Since the number of components translates directly to the number of instructions rather than the size of the circuit, this conversion reduces instrumentation overhead in most modern processors.

For our experiments, we augmented the OpenRISC processor pipeline with an operation profiler in its instruction decode stage. The operation profiler consists of 80 pattern matchers that detect each instruction issued at the instruction decode stage of the processor and 70 pattern matchers that detect activation of peripheral components. All 150 pattern matchers of the profiler have 16-bit counters to keep track of all operations. According to the CAD tool log, above hardware instruments accounts for less than 1% of the total FPGA resource.

This methodology produced a power data that was shown to deviate from the actual power measurement by only 2.85%. This result is 4 to 10 times better than other state-of-the-art methods. Additional details on our experiment setup, the methodology, and the result are in [30].

IV. Mapping Operation to Architectural Components

Our groundwork in energy per operation (EPO) method [30] allowed us to drastically reduce hardware resource over EPC method [26-29] while not sacrificing the accuracy of on-line power monitoring of FPGAs. However, unlike EPC method, EPO result does not directly map to physical components within FPGA design. Therefore, it is difficult to directly determine unique energy consumption rates of different parts of a chip. Therefore, we present and evaluate a new correlation method that automatically translates data collected for EPO method into EPC of SoC without any additional hardware modification.

A. Understanding Soft Microprocessor Interoperation

At a glance, obtaining EPCs of SoC using the same instruction profilers used for EPO seems impossible. However, there exists a close association between SoC operation and component activity because microprocessor control signals that activate components in a SoC.

In a typical microprocessor-based SoC, a software program running on its processor controls all its activities. Each instruction triggers signals that control the activity of different sub-components of a processor and its peripheral. Therefore, it is possible to map every instruction to various sub-components of a microprocessor by examining its architecture.

For our FPGA platform, we had a complete hardware description for the soft microprocessor based SoC named OpenRISC. Therefore, we spent our time to carefully study the code base to hierarchically map each of its instruction to the corresponding microarchitectural components.

At the first level of the hierarchy, the instructions were divided based on their operation types. This classification resulted in seven sets: ALU, Multiply-and-Accumulate, Shift-Rotate, Branch, Load-Store, Compare and Floating-Point Operations. Then, an additional level of components was used to differentiate different power draw across the instructions within each set. By doing this, we not only identified power consumed by the parts of the circuit that are unique to each subset of instructions but the power consumed by the shared parts of the circuit.

Based on this study of OpenRISC architecture, we mapped 80 OpenRISC instructions are to 55 physically identifiable microarchitecture components. Table 2 is an example of this instruction-to-component map (3 levels of sub-components within ALU).

No	Component Name	OpenRISC Instructions
1	ALU	18-23, 53-58, 64
1.1	ALU_ARITH	18-23, 53-58
1.2	ALU_SHROT	25-28, 60–61
1.3	ALU_SPL	65-70
1.4	ALU_SPRS	18-19, 53-55
1.1.1	ALU_ARITH_ADD	18-19, 53-54
1.1.2	ALU_ARITH_AND	20, 56
1.1.3	ALU_ARITH_OR	21, 57
1.1.4	ALU_ARITH_SUB	55
1.1.5	ALU_ARITH_XOR	22, 58

Table 2: OpenRISC Instruction to ALU Sub-component Map

B. *Instructions to Component Activations*

For this task, we made no additional hardware modifications to the OpenRISC SoC that we changed for EPO method. The hardware operation profiler in this SoC counted the executed instructions and peripheral operations at runtime. Then a new software function was used to convert all of the operation counts into mapped sub-component activation counts. More specifically, the new function converted samples of 80 instruction counts into 55 sub-component activation counts by simply adding the instruction count values to mapped hardware sub-component activation count registers. Since other 70 operation counts are directly associated with peripheral SoC components, these counts were directly used as physical component activation counts.

C. *Calibration and Power Extraction*

Based the fundamentals of regression and estimation [7], we can solve for EPCs using operation activation count data, and the associated power measurements collected during a fixed period. For our experiments, we assume that dynamic thermal profile of a chip is the main contributor to any changes in leakage power. Since heat transfer is relatively slow compared to the digital circuit activities, we assume that leakage is relatively stable over a short period.

By converting operation counts to sub-component activation counts, unknown components in Eq. 2 was reduced from 151 down to 126 (125 EPC coefficients of the sub-components and a total leakage power component); thereby drastically reducing computation and memory requirements. For 56 unknown components, the equation can be re-written as the following:

$$P_{meas} = \sum_{j=1}^{125} a_j w_j + L + R \qquad (eq.\ 3)$$

To reduce the computation complexity, we used the *matrix inversion lemma* presented by Mendel [7] to reduce the matrix inversion into an arithmetic division.

Along with this adaptation, we designed a light-weight regressive independent component analysis (ICA [20]) algorithm to minimize R for a given window of sample data. After subsequent EPC computation, the time window of the equation is shifted by a single sample to re-compute the new EPCs through regressive minimization of the residual component of the newly added equation. When this algorithm was applied to sliding time window of actual measured data, we observed that EPOs and leakage values converged to stable values. These stable weights are used by our system to compute power consumed by all of the monitored sub-components.

Since thermal and voltage profile of a chip changes over time, our algorithm can be applied continually to extract the best possible solution under any given condition.

D. *Benchmark Programs*

Since embedded Linux runs on OpenRISC SoC, there are many programs that we can use to evaluate our system. However, we learned that existing benchmark suites for OpenRISC exercised all parts of the SoC. Therefore, in addition to a benchmark suite, we custom wrote a number of programs to activate all of the sub-components and extract their EPCs.

We used Coremark benchmark suite for OpenRISC to exercise several components of the SoC. This benchmark is built to exercise the processor and some of its associated peripherals. In addition to activating all parts of the processor, it exercises the memory DMA, Wishbone BUS, and parts of VGA during its execution. Like the EPO experiments, we created a script to run each of the benchmark kernels 100,000

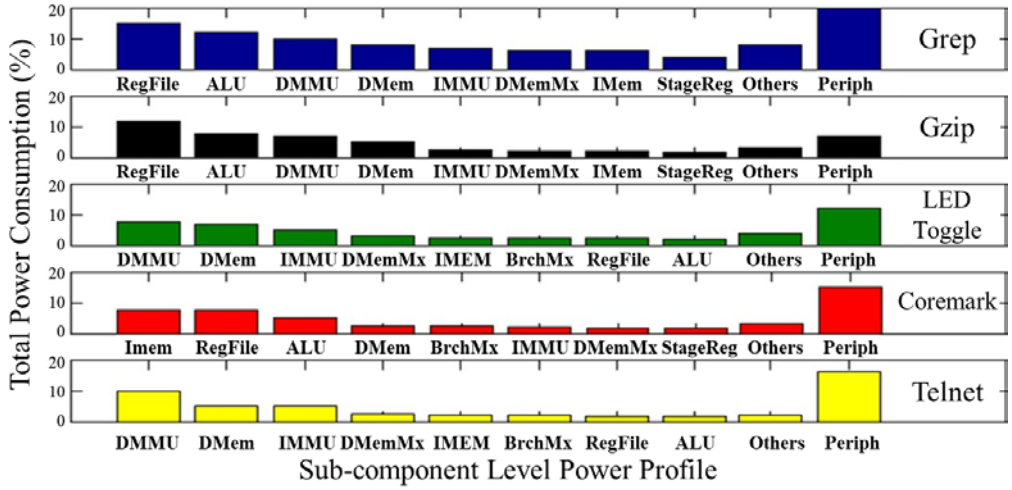

Figure 3: Results run-time EPC measurements for benchmark programs

iterations.

OpenRISC SoC has peripherals including UART, GPIO, and additional DMA engines that were not activated by the Coremark. Therefore, we wrote a Telnet communication program that used Memory, UART, and DMA interface. A separate program was created for GPIO. Since GPIOs have additional circuits to drive larger external loads, we created a program that used the GPIO and the Interrupt interface to toggle an external light-emitting-diode (LED).

Lastly, we included two commonly used applications for Linux, grep and gzip. We included these programs because they were computationally intensive and designed to exercise several components of the SoC during its execution.

E. Experimental Result

As with EPO experiments, data for operation counts and power measurements were collected for calibration and computation of energy. However, an extra processing step was added to convert operation counts into sub-component activation counts. This converted data was used in eq. 3 to extract EPC for all of 125 components, total leakage power, and residual power.

Then the EPCs were multiplied with the sub-component activity counts to compute precise power distribution of sub-components of the design during the execution of the test bench. Figure 3 shows the power distribution profile result listing top eight sub-components, listed in the order of highest to lowest power weight values, within the OpenRISC processor core, the sum of peripheral component power, and the sum of the miscellaneous component power of the SoC.

V. Data Analysis

While analyzing our experimental data, we observed interesting issues and solutions that system designers may consider for the best system performance and cost.

A. Time Synchronization

In a microprocessor-based SoC, it is often difficult to precisely synchronize sampling of the internal digital sensor and a direct power measurement data due to non-deterministic behaviors of software interrupts and queuing delays. Because of this uncertainty, we needed a way to adjust the alignment of the power data and the component activation count data.

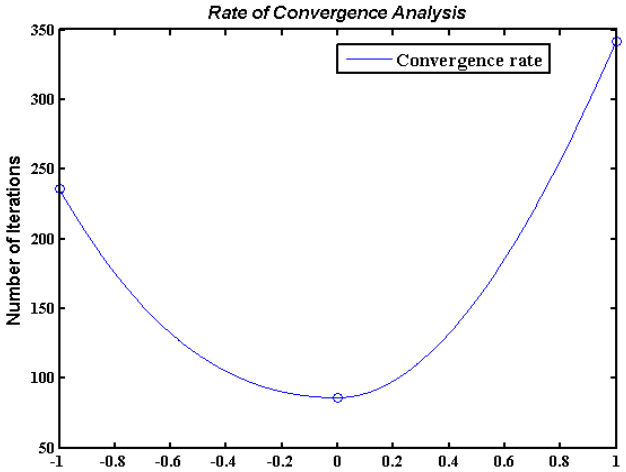

Figure 4: Number of Iteration required for EPC convergence versus timestamp offset between power data and component count

During our implementation process, we found that changing time delay of the external power monitoring instrument was not only difficult but imprecise. Instead, we decided to change our digital sensors to sample at a much higher sampling rate than that of power measurements. Given this higher resolution operation count data, we were able to not only compute the lower resolution count data that matched the sampling rate of the power measurements but also change the time alignment of the data by changing the grouping of the higher resolution samples for the computation.

This flexibility allowed us to iteratively shift the time alignment for operation count data before applying our regressive algorithm on the data and power measurements to extract EPCs. One interesting observation of the iteration data shown in Figure 4 is that the EPC values for all of the tested shift alignment converged to the same value. But more importantly, we found that the best time alignment caused the EPC values to converge the fastest.

This one-time software-based method allowed us to eliminate the need to time synchronize these sets of data using specialized hardware.

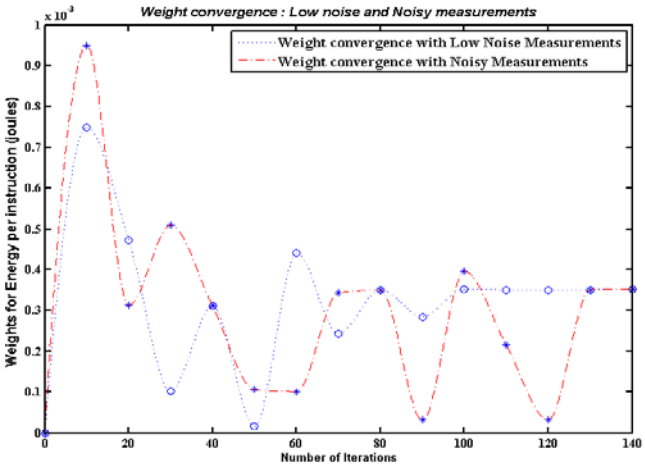

Figure 5: EPC convergence rates given different ADC measurements

B. Noise Suppression

Current sense resistor with a signal amplifier and ADC may be considered the best method of obtaining the ground truth power measurement. However, it has some practical challenges including that of the trade-off between cost/sampling rate/accuracy, calibration error in resistors, the effect of aging, and noise introduced by instrumentation components.

To evaluate the impact of differences in power measurement quality, we design an experiment with two different external ADCs collecting power measurements for the same FPGA. The first ADC (ADC_1) was a higher quality ADC with the sampling rate of 5 KHz. The second ADC (ADC_2) was a much lower quality ADC with a slower sampling rate of 1 KHz.

As expected, the convergence rate for the EPCs using higher quality measurements was, indeed, faster than the lower quality measurements as shown in Figure 5. However, we found that both results converged to the same EPC values for all of the component.

We observed that our algorithm's ability to suppress noise numerically allowed the system to produce an accurate final result despite the differences in sampling rates and quality of power measurements. In practice, this difference in calibration time needed for convergence is so small that using higher quality measurement instrument may only incur higher resource cost without any gain in measurement accuracy.

C. Measurement Accuracy

The EPO method assigns a single instruction count value to summarize various component utilization. While this method proves to work sufficiently well, it leaves a small room for measurement error by ignoring physical aspects of the system. In other words, while there are physical components shared between multiple instructions, EPO method merges all of the power consumed by the components into a single value and effectively making each of the components as separate entities even though many of the components used by different operations. Because of this, any accidental correlation between noise and operation execution pattern would cause EPO result to absorb the noise. This error, in turn, negatively affects the rest of the EPO computation. On the other hand, identifying physical components for each operation and grouping activity counts of components reduces the number of unknown components and strengthens the noise filtering function of the algorithm.

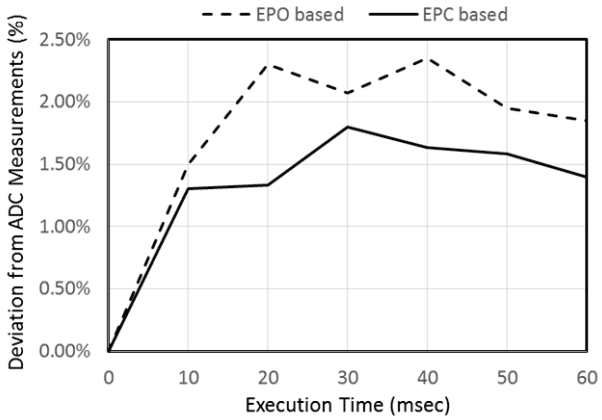

Figure 6: Power deviation plot for the Dhrystone benchmark

To quantify this deviation, we used the same dataset collected for EPO work for EPC experiment. Figure 6 shows a deviation plot of the total power values derived from extracted EPO and EPC from external power measurements while executing Dhrystone benchmark. For this benchmark, we found that the maximum deviation of power derived from EPO method was at about 2.35% from direct measurements while EPC method yielded a closer result with 1.80% maximum deviation. Similarly, we compared the power results against the directly measured power data for all of the benchmark programs and found that the maximum deviation for EPO method was approximately 2.85% while EPC method produced better results with the maximum deviation of 1.85%.

VI. Related Works

The FPGA manufacturers provide simulation-based power estimation techniques that use a pre-evaluated library of power utilization for various devices. These libraries consist of power usage by different types of lookup tables and other on-chip analog components such as phase-locked-loop (PLL). This information is used along with the lookup table and component usage on the target FPGA by the user designs to estimate power.

In this estimation technique, an initial offline capacitive power model is built based on total power measurements and event sets [3-5,8-10,12-16]. A power model is built associating total current drawn by the platform and performance counter values that indicate various events on the processor. Further, Iyer et al. [11] build a system that updates this power model on-chip before its first use by the power management unit.

In this technique, direct current measurements or values from simulations are used to derive average power dissipated per instruction [12,17-19]. The energy per instruction is modeled using offline regression analysis based on the collected power data, access rates, execution times and instruction counts. These values are then used to estimate the energy consumed by a program at runtime [21-24].

The related works mentioned in this paper share the core equation of our original work sub-component level power monitoring [26-28]. While we demonstrated all of our works using Xilinx FPGA based platforms, another group of researchers has recently produced a result on Altera FPGA based platforms with similar accuracy and resolution using an algorithm based on the same underlying theory and methodology; verifying that the approach is applicable across different platforms [25].

However, we deliberately point out that our contribution in this paper is the method of architectural association, data conversion, and modified use of digital counters. Also, we integrated software interface in a commodity Operating System that allowed real-time access and processing of the measurements using a user-level application.

VII. Conclusion

This paper presents a low overhead hardware/software co-designed method for measuring power consumed by sub-components of FPGA-based SoCs. We used compact hardware sensors in instruction decode stage of a processor to collect executed instruction counts. Then we converted the instruction execution patterns into physical component activities within the SoC during runtime. Using this data and the total chip power measurements, we presented an automated method that successfully extracted accurate EPC. Through several benchmark experiments, we showed that our method could measure all the subcomponent power at a higher sampling rate than other methods we surveyed. Furthermore, the experiments show that our results are closer to the ground truth than the other traditional methods. The in-depth comparison in Table 3 reveals additional advantages in our method over others. However, we find that the biggest strength of our method is an accurate on-line calibration using fine-grained on-chip digital sensors and an external power sensor.

We contend that such accurate but practical method of online component level power measurements will provide the means for future adaptive circuit designs to function optimally even in the presence of worsening PVT variations found in current and emerging fabrication technologies.

REFERENCES

[1] V. De, "Fine-Grain Power Management in Microprocessors," ISSCC 2013.

[2] M. Pedram, J. Rabaey, "Power-Aware Design Methodologies," Kluwer Academic Publishers, 2002.

[3] K. R. Stokke, "High-Precision Power Modelling of the Tegra K1 Variable SMP Processor Architecture," IEEE Symp. On Embedded Multicore/Many-core Systems-on-Chip (MCSoC), France, Sept. 2016.

[4] D.Brooks et al. "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations." ISCA-27, June 2000.

[5] C. Isci and M. Martonosi, "Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data," Proc. MICRO, 2003, pp. 93-104.

[6] M.Pedram, and H. Vaishnav. "Power optimization in VLSI layout: a survey." Journal of VLSI signal processing systems for signal, image and video technology 15.3 (1997): 221-232.

[7] J.Mendel, "Lessons in Estimation Theory for Signal Processing, Communication and Control," Prentice-Hall, Englewood-Cliffs, NJ, 1995.

[8] V. Tiwari et al., "Instruction Level Power Analysis And Optimization Of Software." In Technologies for wireless computing, pp. 139-154, Springer, 1996.

[9] P. Landman, J. Rabaey, "Architectural Power Analysis: The Dual Bit Type Method," IEEE TVLSI, pp. 173-187, June 1995.

[10] W. Wu et al. "A Systematic Method For Functional Unit Power Estimation In Microprocessors," In Proceedings of the 43rd DAC, pp. 554-557, July 2006.

[11] A. Iyer, D. Marculescu, "Power Aware Microarchitecture Resource Scaling," Proceedings of the conference on DATE, 2001.

[12] S. Lee et al., "An Accurate Instruction-Level Energy Consumption Model For Embedded RISC Processors," ACM SIGPLAN Notices. Vol. 36. No. 8. ACM, 2001.

[13] P. Kalla et al., "SEA: Fast Power Estimation For Microarchitectures," Proceedings of the 2003 ASP-DAC Conference, 2003.

[14] A. Carroll, H. Gernot, "An Analysis of Power Consumption in a Smartphone," USENIX Annual Technical Conference. 2010.

[15] S. Hong, H. Kim, "An Integrated GPU Power And Performance Model," ACM SIGARCH Computer Architecture News. Vol. 38. No. 3. ACM, 2010.

[16] J. Haid et al., "Run-Time Energy Estimation In System-On-A-Chip Designs," In Proceedings of the 2003 ASP-DAC, pp. 595-599, January 2003.

[17] Y. S. Shao, D. Brooks, "Energy Characterization And Instruction-Level Energy Model Of Intel's Xeon Phi Processor," In Proceedings of the ISLPED, pp. 389-394, September 2013.

[18] A. Annamalai et al., "Reducing Energy per Instruction via Dynamic Resource Allocation and Voltage and Frequency Adaptation in Asymmetric Multicores," In ISVLSI, pp. 436-441, July 014.

[19] W. J. Song, W et al., "Instruction-based energy estimation methodology for asymmetric manycore processor simulations." In Proceedings of the 5th International Conference on Simulation Tools and Techniques, pp. 166-171, March 2012.

[20] A. Hyvarinen et al., "Independent Component Analysis, " John Wiley and Sons, New York, 2001.

[21] E. Handschin et al., "Bad Data Analysis For Power System State Estimation," IEEE Transactions on Power Apparatus and Systems, 94(2), 329-337, 1975.

[22] F. C. Schweppe, "Power System Static-State Estimation, Part III: Implementation," IEEE Transactions on Power Apparatus and Systems, (1), 130-135, 1970.

[23] V. Tiwari et al., " Instruction Level Power Analysis And Optimization Of Software," In Technologies For Wireless Computing, pp. 139-154, Springer, 1996.

[24] H. Mehta et al., "Accurate Estimation Of Combinational Circuit Activity." In Proceedings Of The 32nd Annual ACM/IEEE DAC, pp. 618-622, January 1995.

[25] K E. Hung, et. al., "KAPow: A System Identification Approach to Online Per-module Power Estimation in FPGA Designs," in Field-Programmable Custom Computing Machines (FCCM), 2016 IEEE 24th Annual International Symposium on, 2016, pp. 56–63.

[26] Y. H. Cho, S. Bhargav, A. Goodney, "Digital Circuit Power Measurements using Numerical Analysis," US Patent 9,618,547 B2, January 24, 2012 (issued April 11, 2017).

[27] S. Bhargav, Y. H. Cho, "Measuring Power Digitally with Numerical Analysis," ACM SIGMETRICS, London, UK, June 2012.

[28] Y. H. Cho, S. Bhargav, A. Goodney, "Digital Signal Transition Counters for Digital Integrated Circuits," US Patent Application 14/226,085, March 26, 2014.

[29] S. Bhargav, Y. H. Cho, "Accurate Power Measurement Technique for Digital Systems using Independent Component Analysis," IEEE DCIS 2015, Estoril, Portugal, November 25-27, 2015.

[30] S. Bhargav, R. K. Prabakar, Y. H. Cho, "Accurate In-situ Runtime Measurement of Energy per Operation of System-on-Chip on FPGA," IEEE Reconfig 2015, Mexico, December 2015.