# All-Digital Embedded Meters for On-line Power Estimation

Daniele Jahier Pagliari, Valentino Peluso, Yukai Chen, Andrea Calimera, Enrico Macii and Massimo Poncino
Dipartimento di Automatica e Informatica, Politecnico di Torino, Turin, ITALY
Email: {daniele.jahier,valentino.peluso,yukai.chen,andrea.calimera,enrico.macii,massimo.poncino}@polito.it

***Abstract*—Modern low power designs use multiple knobs for concurrent dynamic and leakage power optimization; supply voltage and threshold voltage are the most adopted. An efficient control of these knobs needs management policies aware of the power breakdown. This implies the availability of smart on-chip strategies for dynamic and leakage power estimation at runtime. In this paper, we address this issue proposing the implementation of embedded dynamic/static power meters that use an optimized regression model fed with data collected from in-situ activity monitors. The number of sensors, their bitwidth and optimal placement are obtained through an automated design flow. The methodology works for general logic and applies not just to processor cores, but also to application-specific designs.**
**We apply our solution to a representative class of benchmarks, showing that it can achieve an average estimation error smaller than 3%, with limited area and power overheads.**

## I. Introduction and Motivation

Dynamic Voltage-Frequency Scaling (DVFS) and Dynamic Threshold Voltage Scaling (DVTS) are key power management techniques that have featured prominently in multi-core SoC designs. Originally devised as orthogonal low-power strategies, their use on the same chip may give substantial boost to the energy efficiency. Several works have modeled and quantified the potential advantages of combining the two [1]–[4]. Nevertheless, a practical solution for their on-chip integration does not exist yet.

DVFS exploits the cubic dependence of dynamic power $P_{dyn}$ on $V_{dd}$ and $f_{clk}$, while DVTS leverages the exponential relationship of leakage power $P_{leak}$ with $V_{th}$. Although they operate on distinct metrics using independent knobs, both have the same effect of increasing circuit delay. An efficient integration of the two implies the use of a "smart" power manager able to decide, at run-time, whether it is better to sacrifice performance by scaling $V_{dd}$, thereby giving priority to dynamic power, or $V_{th}$, thereby giving priority to static power. Making the "right" decision implies the knowledge of how dynamic and static consumption contribute to the total power at any instant of time [2], [3].

A given design may in fact experience periods in which dynamic power is predominant, and others in which leakage does. The former are commonly associated to high switching rates, high frequency and high supply voltage; the latter, by contrast, to low utilization rates, low frequency and low voltage. However, it is hard to identify a crisp boundary between the two scenarios. Modern SoCs work under a ultra-wide voltage range (50% of nominal $V_{dd}$ is a common practice), and may show high activity even at low frequency [5]. Moreover, the relative dominance between dynamic and leakage consumption may drift from its nominal value due to process-variation and temperature, that occur only at post-silicon [6]. The preliminary conclusion is that a concurrent use of DVFS and DVTS can be enabled only through on-chip "sensors" that measure $P_{leak}$ and $P_{dyn}$, separately, and with a reasonable degree of accuracy. Unfortunately, there is no hardware or software solution able to derive this information; it is no coincidence that previous methods in [1]–[4] assume a pre-defined ratio between $P_{leak}$ and $P_{dyn}$.

This paper aims at filling this gap introducing the design and optimization of a new class of *embedded power meters for dynamic and leakage power estimation at runtime*. The proposed method is fully automated, and can be applied to generic digital circuits, from general purpose cores to application specific accelerators. It is based on regression models learned at design time and inferred on-chip using data collected from in-situ activity monitors. Two different versions of power meters are presented and characterized, a *software-based* version, well suited for SoCs with a programmable micro-architecture, and a *hardware-based* one, tailored to fit the specs of custom circuits. Both show low area overheads (1.45% for the SW-based version, 7.14% for the HW-based one, on a small RISC core [21]) ensuring an average estimation error smaller that 3% over a wide range of PVT conditions.

## II. Background and Related Work

### A. Background

The power consumption of a digital circuit is composed of dynamic and static contributions [7]. Dynamic power (i.e., consumed during transistors switching) is mostly due to the charging and discharging of transistor capacitances, and for a single device can be modeled as [3], [6]:

$$P_{dyn} = \alpha \cdot C_L \cdot f_{clk} \cdot V_{dd}^2 \tag{1}$$

where $f_{clk}$ is the clock frequency, $V_{dd}$ is the power supply voltage, $C_L$ is the equivalent load capacitance of the device, and $\alpha$ is the *activity factor*, i.e. the probability of the output making a transition in each clock cycle.

Static power consumption is due to leakage currents originating from a number of physical phenomena [7]. In high-$k$ dielectric devices, the most relevant source of leakage is subthreshold conduction, which can be modeled as [6], [7]:

$$P_{leak} \propto \beta \cdot V_{dd} \cdot e^{-\frac{V_{th}}{\gamma \cdot V_T}} \tag{2}$$

where $V_{th}$ is the threshold voltage, $V_T$ is the thermal voltage, and $\gamma$ and $\beta$ are technology-dependent parameters.

978-3-9819263-0-9/DATE18/©2018 EDAA

Dynamic Voltage and Frequency Scaling (DVFS), a standard for digital circuits [6], [8]–[10], reduces both $P_{dyn}$ and $P_{leak}$ by lowering $V_{dd}$ to the minimum value that ensures timing correctness for a given clock frequency $f_{clk}$; the latter is determined depending on the system workload.

Leakage power can be controlled through Dynamic $V_{th}$ Scaling (DVTS), a technique that leverages the exponential dependence of $P_{leak}$ on transistors' threshold voltage [11]; similarly to DVFS, $V_{th}$ is tuned till reaching timing compliance with $f_{clk}$ as constraint. DVTS is traditionally implemented in bulk CMOS technology using body bias, i.e. polarization of the substrate terminal [1], [11]. In modern Fully-Depleted Silicon-On-Insulator (FDSOI) technologies [6], [12], however, the polarization of the back-gate terminal of transistors has a stronger effect on $V_{th}$, reaching a sensitivity (*body factor*) as high as $85\,\mathrm{mV/V}$ [12]. Moreover, the range of polarization voltages that can be applied is also higher than in bulk.

### B. Related Work

Achieving accurate estimation of the breakdown of total power into static and dynamic components is not trivial. To the best of our knowledge, there is no hardware or software solution able to derive this information.

Direct supply current measurements such as [13] require a dedicated analog meter, an uncommon option for general digital SoCs, which, however, returns only the *total* power.

Fully-digital solutions for core-based systems [14]–[16] and GPUs [17] make use of *performance counters*; these methods establish an empirical correlation between power consumption and the occurrence of specific events, e.g., cache accesses, number of fetched instructions, etc. [15]. As for analog meters, only the *total power* is measured. Moreover, this approach is core-based, hence unpractical for fine-grain DVFS/DVTS [18] or for custom blocks devoid of performance counters.

## III. Proposed Methodology

The objective of this work is to devise on-chip dynamic and leakage power meters that can be used in any digital circuit. Their design is supported by few basic assumptions. Firstly, we assume that the values of $V_{DD}$, $f_{clk}$ and of the body/back-bias voltages for NMOS and PMOS transistors ($V_{bb,n}$ and $V_{bb,p}$ respectively), are known at all times. This is reasonable, as these are exactly the knobs used for power management, and their value is commonly stored in registers. Without loss of generality, we assume $V_{bb,n} = -V_{bb,p} = V_{bb}$. Secondly, we assume that the power management unit gets thermal information using distributed on-chip temperature sensors [19], which have an error in the order of few °C.

### A. Dynamic Power Estimator

The total dynamic power consumption of a circuit is obtained summing the contribution of every logic gate using (1):

$$P_{dyn,tot} = \left( \sum_{i=0}^{N_{gates}} \alpha_i C_{L,i} \right) f_{clk} V_{dd}^2 = C_{eff} f_{clk} V_{dd}^2 \qquad (3)$$

where $C_{eff}$ is the *effective capacitance*. Given the previously mentioned assumptions, the problem of estimating $P_{dyn}$ reduces to estimating $C_{eff}$.

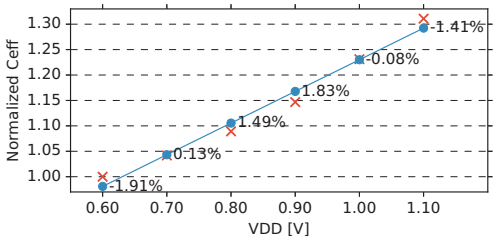


Fig. 1. $C_{eff}$ versus $V_{dd}$ for the RI5CY [21] core.

The $C_{eff}$, in principle, depends on the activity factor of every gate in the circuit. However, a key characteristic of the proposed metering solution is that $C_{eff}$ **can be accurately estimated monitoring the switching activities of a small subset of significant Flip Flops** (FF) and then combining them via linear regression. In fact, gates activities are often strongly *correlated* with each other, because of constraints imposed by circuit topology. Thus, sampling a small subset of key points is sufficient to estimate the overall activity.

Building the aforementioned regression model requires *training data*, i.e. estimated activity/power values for a set of representative workloads, collected through back-annotated gate-level simulations. Then, a *feature selection* method can be used to identify the key FFs, based on their influence on $C_{eff}$. Besides being applicable to general circuits, this approach also permits a potentially better tracking of power, with respect to methods based on performance counters. Indeed, the latter only monitor high-level system events, while we allow significant events to be sampled at *any location* in the circuit. We adopt a simple, yet effective feature selection approach [20] described as follows. First, a *L1-regularized* linear regression model (called *Lasso*) is constructed, considering the switching probability of all FFs. Due to regularization, this model is sparse, i.e. contains a large number of coefficients at 0. To extract the $N_{FF}$ most relevant FFs, it is sufficient to select the features corresponding to the largest coefficients [20]. Using realistic workloads during the training stage allows a substantial reduction of the number of FFs required to obtain a good estimation accuracy, since the model will be built only based on combinations of $\alpha_i$ that really occur during the operation of the design. Thus, in order to make the model reliable, the training set should represent as much as possible the full spectrum of tasks executed by the system.

Due to the presence of short-circuit currents and other parasitic effects lumped into $C_{eff}$, this quantity also depends on the operating conditions of the circuit. A first order dependence is with $V_{dd}$ [7]. Although non linear, this relation can be safely linearized in normal operating conditions (e.g. $V_{dd} \in [0.60\,\mathrm{V}, 1.10\,\mathrm{V}]$ for the technology considered in Section IV). This is shown in Figure 1, where the measured dependency and the corresponding regression errors are plotted for one of our reference benchmarks, and for a generic workload. Thanks to linearization, $V_{dd}$ can be directly added to the regression model, which results as follows:

$$C_{eff} \simeq w_0 + \sum_{i=1}^{N_{FF}} \alpha_i w_i + V_{dd} w_{N_{FF}+1} \qquad (4)$$

where $w_i$ are regression weights.

The presence of $V_{dd}$ imposes that the set of training data contains simulations obtained for all the relevant supply voltages. Notice that $C_{eff}$ secondarily depends on temperature and body/back-bias voltages. In this work we treat these dependencies as negligible. However, our methodology can be easily extended to these second-order variables, using the polynomial approximation method described in the next section to deal with non-linear relations.

### B. Leakage Power Estimator

Leakage power does not depend on switching probability, but is affected by the *static probability* of signals fed to the inputs of logic gates [7]. However, in a large-enough circuit this dependence is averaged over the different gates, thus making leakage substantially independent on workload. We empirically proved this assumption executing 150 different workloads on the RI5CY core [21] (see Section IV): results show a worst-case deviation of $P_{leak}$ from its average value of $\approx 1\%$. Conversely, $P_{leak}$ is heavily influenced by temperature ($T$), $V_{dd}$ and $V_{th}$, the latter affected by $V_{bb}$ [6]. Most of these dependencies are non-linear, as evident from (2).

Similar to what is done for $P_{dyn}$, we adopt a regression-based approach to estimate $P_{leak}$ at runtime; a high-order polynomial model is needed due to the aforementioned non-linearity. Since polynomials are still computed through additions and multiplications, similar HW can be used to evaluate both $P_{dyn}$ and $P_{leak}$ (see Section III-C).

The regression model for $P_{leak}$ is as follows:

$$P_{leak} = \sum_{i=0}^{d} \sum_{j=0}^{d-i} \sum_{k=0}^{d-i-j} w_{i,j,k} V_{dd}^{i} \cdot V_{bb}^{j} \cdot T^{k} \qquad (5)$$

where $d$ is the polynomial model degree. Moreover, we further reduce complexity applying the same feature selection method described in Section III-A, to identify the terms of (5) that are best correlated with $P_{leak}$. We refer to $N_{leak}$ as the number of selected features for leakage estimation.

Differently from the case of $P_{dyn}$, the *selected features* will be independent from the circuit, as they are environmental conditions. What will change from one circuit to the other are only the regression weights $w_{i,j,k}$. Concerning the training set, since $P_{leak}$ is not affected by internal switching activities, data can be collected from simulations performed on idle circuits under different $T$, $V_{dd}$, and $V_{bb}$.

### C. Hardware Implementation

The $P_{dyn}$ regression model needs a measurement of the switching probability $\alpha_i$ of key FFs, which can only be obtained through hardware instrumentation. An approximation of the switching probability can be obtained simply counting the number of toggles at the output of a FF over a fixed number of clock cycles (i.e. the *toggle rate*). A basic switching probability *meter* can thus be implemented in hardware as shown in Figure 2, for an example with $N_{FF} = 3$.

The circuit consists of an array of XOR gates which perform a parity check between the key FFs and their shadow copies; a logic-1 at the output of the XORs implies the key FF switched. In that event the corresponding counter is incremented. A

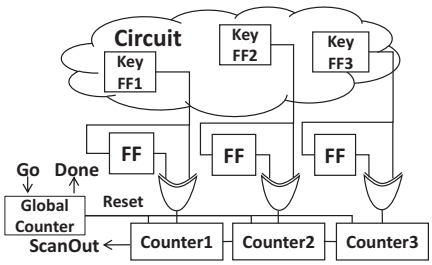


Fig. 2. Meter hardware, software-based version.

global counter is used to trigger the estimation intervals, namely, to synchronize the reset of each FF counter at the beginning of every new estimation interval; the same global counter is in charge of delivering the *Done* signal when the correct number of clock cycles has elapsed.

All counters have the same bit-width $M$; therefore, a new estimate is produced every $2^M$ cycles. The switching probability for the $i$-th FF is simply computed as $Counter_i/2^M$. Notice that since the estimation period is a power of 2, divisions can be replaced by shifts. The final count values are serially scanned out in $M \cdot N_{FF}$ cycles.

When using this simple meter, the final estimates of $P_{dyn}$ and $P_{leak}$ must be computed externally via software. In most systems, this is not an issue. In fact, even in heterogeneous SoCs including accelerators, power management is normally already implemented in software, either on the main processor or on a dedicated microcontroller. In these cases, this *software-based* meter is preferable, due to its smaller overheads.

However, it is also possible to implement the complete models of (4) and (5) fully in hardware. This requires some additional circuitry, sketched in Figure 3. The basic additional component is a Multiply-and-Accumulate (MAC) unit, with an intermediate product register. This block can be reused for both dynamic and leakage estimates, thanks to the fact that both models are based on (polynomial) linear regression. The MAC inputs are selected via two multiplexers, driven by a Control Unit (CU). The CU schedules calculations to produce either $P_{leak}$ or $P_{dyn}$ estimates, depending on an input command signal. To compute high order polynomial terms in (5), the multiplier output can be fed back to the multiplexers. This implementation allows *autonomous power estimation* for any generic circuit, without any support from external CPUs.

If compared to the software-based version, the *hardware-based* meter takes some additional area and hence some extra power consumption. In order to minimize the overheads, arithmetic operations are performed in fixed-point; this requires a conversion of the regression weights from their native floating-point representation. This issue is managed through the automated design flow described in the next section.

In both meter versions, the counters bit-width $M$ is a key design parameter. First, it affects the *resolution* of the switching probability measurement (the bigger $M$, the better). For instance, a 2-bit counter can only produce 4 toggle rate estimates (i.e. 0, 0.25, 0.5 and 0.75). Second, it determines the *estimation time*, by affecting the switching probability measurement duration (in both versions), and the scan-out phase (in the software-based) or the computation phase (in the

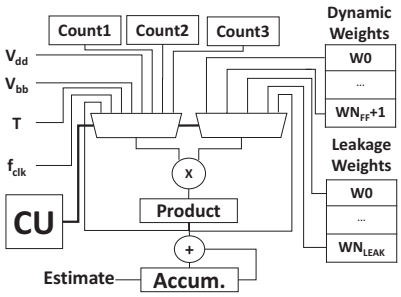


Fig. 3. Meter hardware, additional datapath for hardware-based version.

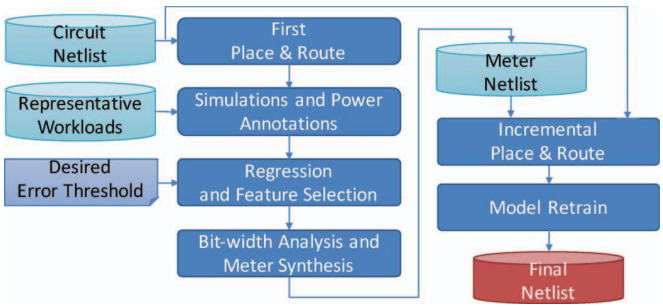


Fig. 4. Implementation flow diagram.

hardware-based). The latter is also influenced by the number and degree of the $N_{leak}$ leakage power features. Finally, $M$ also impacts the area and power overheads of the meters.

### D. Implementation Flow

We have implemented an automatic tool to instrument circuits obtained from a standard design flow with the proposed power meters. The execution flow, shown in Figure 4, makes use of industrial EDA software for synthesis and Place and Route (P&R). It takes as inputs the original circuit netlist, a set of representative workloads (e.g. testbench vectors, compiled programs, etc), and a user-imposed threshold that defines the allowed average estimation error. The type of meter (software-based or hardware-based) must also be selected. The final output is a placed & routed design instrumented with the meter. The main steps of the flow are the following:

**1) First P&R**: a standard P&R is run on the original circuit.

**2) Simulations and power annotations**: the post-P&R design is simulated with the representative workloads for all the $V_{dd}$ values. Then, a timing and power analysis is performed to annotate switching activity and $P_{dyn}$ values.

**3) Regression and feature selection**: workload data collected from simulations are fed to the regression model and feature selection algorithm for $P_{dyn}$ described in Section III-A. To avoid overfitting, workload data is divided into multiple training and test splits. Model coefficients are built using the training set, whereas the test set is used for evaluating the estimation error. In this phase, key FFs for $P_{dyn}$ estimation are identified. Leakage features, instead, are selected based on the imposed error threshold, regardless of the switching activity, as explained in Section III-B. The tool selects the minimum number of features that achieves the user-defined error threshold, with some additional margin ($0.2\%$) to compensate approximations due to fixed-point conversion.

**4) Bitwidth analysis and meter synthesis**: once the features are selected, the tool executes an accuracy analysis to select the appropriate bitwidths of the counters and (in the case of the HW-based meter) of the fixed-point datapath. Then, the meter netlist is automatically synthesized from a generic template.

**5) Incremental placement**: key FFs in the original design are connected to the meter inputs, and an incremental P&R is performed on the whole circuit. This lets the P&R tool place the meter circuitry, freely, thus achieving timing compliance and minimum impact on area and power consumption.

**6) Final retrain**: after the circuit has been instrumented with the meter, the coefficients of the two power models are recomputed. For $P_{dyn}$, this allows to account for variations in power caused by the insertion of the meter itself. For $P_{leak}$, it allows tuning the regression weights to the target circuit. Ideally, this phase should be executed *after fabrication*, using real power measurements with testing instrumentation, instead of simulation results. In such case, the final retrain would also allow to account for variability in the manufacturing process.

## IV. Experimental Results

We tested our methodology for dynamic power estimation on three benchmark circuits: a 30-stage 16-bit (Finite Impulse Response) FIR filter, a 256-bit Advanced Encryption Standard (AES) cryptographic accelerator (both available on *OpenCores*), and the open-source RI5CY core [21]. Workload simulations were executed in MGS Questasim 10.6. For synthesis and place and route, we used Synopsys DC and ICC, version L-2016.03, respectively, targeting a 28nm UTBB FDSOI technology from STM. Timing and power analyses were performed with Synopsys PT L-2016.06. The nominal $V_{dd}$ for implementation was set to $0.6\,\mathrm{V}$. Workload simulations were executed in six different $V_{dd}$ conditions, from $0.6\,\mathrm{V}$ to $1.1\,\mathrm{V}$ in steps of $0.1\,\mathrm{V}$. For each $V_{dd}$, the clock frequency was set to the largest value that did not cause timing violations.

Leakage estimation experiments were performed in SPICE (Synopsys CustomSim L-2016.06), in order to consider more $(V_{dd}, V_{bb}, T)$ corners than those available to us as Liberty format characterizations. Namely, we considered: $V_{dd} \in [0.6\,\mathrm{V}, 1.1\,\mathrm{V}]$ with steps of $0.025\,\mathrm{V}$, $V_{bb} \in [-0.3\,\mathrm{V}, 1.1\,\mathrm{V}]$, with steps of $0.1\,\mathrm{V}$, and $T \in [-40\,^{\circ}\mathrm{C}, 125\,^{\circ}\mathrm{C}]$ in 10 uniform steps. Since leakage power does not depend on workload, we used an 11-stage ring oscillator as representative benchmark.

Regression and feature selections algorithms were implemented using Python 3.5 and the *sklearn* library. We used the Mean Absolute Error (MAE) as target error metric. For both $P_{dyn}$ and $P_{leak}$ the MAE has been normalized with respect to the variation of power over the corresponding datasets.

To the best of our knowledge, there are no existing methods for runtime on-chip estimation of separate $P_{dyn}$ and $P_{leak}$ in generic digital circuits. Therefore, we cannot directly compare our method to the state-of-the-art. Nonetheless, for the processor core benchmark, it is possible at least to have a qualitative comparison with the accuracy obtained by methods relying on performance counters. However, this comparison is very rough, since those methods were tested on different architectures, and they aim at estimating total power and not its breakdown in dynamic and leakage [14]–[17].

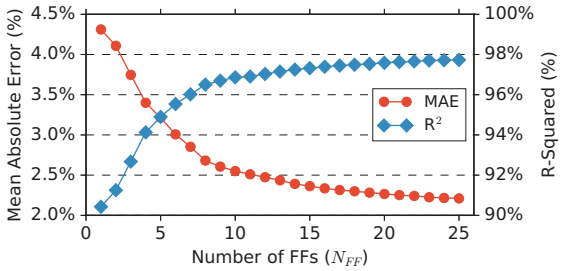


Fig. 5. $P_{dyn}$ estimation error (MAE) and $R^2$ coefficient versus number of key FFs ($N_{FF}$) for the RI5CY core.

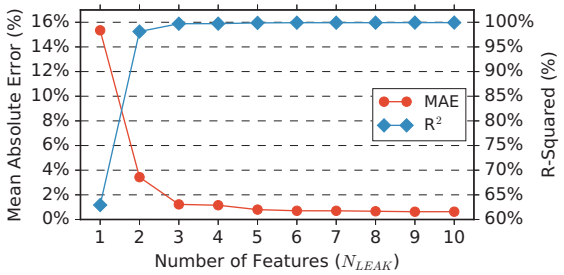


Fig. 6. $P_{leak}$ estimation error (MAE) and $R^2$ coefficient versus number of features ($N_{leak}$) for a 11-stage ring oscillator. Model degree $d = 6$.

### A. Accuracy versus number of features

Figure 5 shows the MAE and the $R^2$ coefficient of the $P_{dyn}$ estimation model, computed using all workloads and $V_{dd}$ conditions, as a function of the number of Key FFs identified by feature selection. The figure refers to the RI5CY core, which contains a total of 2199 FFs. Expectedly, the higher $N_{FF}$, the more accurate the model. However, even a very small number of features (compared to the total number of FFs) is sufficient to achieve good accuracy, as shown by the low MAE values and $R^2 \approx 100\%$.

An analogous plot is shown in Figure 6 for $P_{leak}$ estimation on the ring oscillator. The full model has been constructed setting the maximum polynomial degree to $d = 6$. With this value, the total number of features, i.e. polynomial combinations of $V_{dd}$, $V_{bb}$ and $T$ would be 83. However, as before, only a few of these features suffice to achieve very low error. Almost identical trends were obtained considering ring oscillators with mixed-type gates having different sizes and fan-in.

These results confirm the soundness of the proposed method, proving that regression with few input variables can provide an easily computable yet accurate estimate of both power components. For comparison, methods based on performance counters for total power estimation in [16] and [17] report average errors of 3%-3.9% and 4.7%, respectively.

As a visual example of the final meter performance, Figure 7 compares, for all workloads of the RI5CY core, $P_{dyn}$ results obtained from power analysis in PrimeTime, and those estimated with our regression method. For space reasons, only results for $V_{dd} = 0.9\,\mathrm{V}$ are shown. Workloads are sorted by increasing power for visualization, and y-axis values are normalized to the maximum power over all workloads. This plot confirms that, despite some noise, our estimator is able to predict the variation of $P_{dyn}$ very accurately.

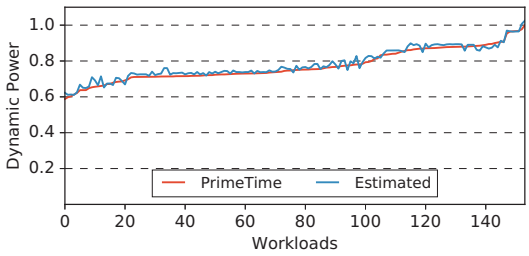


Fig. 7. Estimated $P_{dyn}$ versus PrimeTime output for the RI5CY core for all workloads, at $V_{dd} = 0.90V$, with $N_{FF} = 8$.

### B. Power and area overheads

Although Figures 5 and 6 show that the proposed regression models are accurate in principle, their applicability on-chip depends on the overheads associated with meters. To assess these overheads, we ran the full flow of Section III-D on the FIR, AES and RI5CY benchmarks. In all three cases, we set the target error threshold to $MAE = 3\%$, and we repeated the flow twice first adding a software-based meter (i.e. only the hardware of Figure 2), and then a hardware-based one (i.e. with the additional circuitry of Figure 3).

Detailed results for both types of meters are reported in Table I. The first columns of the table report the number of key FFs selected for $P_{dyn}$ estimation, and the corresponding counters bit-width ($M$). The parameters for leakage estimation are the same for all circuits, i.e. $d = 6$, $N_{leak} = 3$.

Hardware-based (HW) meters generally require a higher bit-width than software-based versions (SW), because of the additional errors introduced by fixed-point calculations in the hardware of Figure 3. Interestingly, RI5CY and AES require a smaller number of features compared to the FIR, despite being larger circuits. This is due to the presence of *control bits* (e.g. the instruction register bits in the RI5CY), whose switching determines the activation of entire datapath sub-blocks. These bits strongly affect the global activity of the design, hence allowing the model to predict $P_{dyn}$ with fewer counters. In general, the higher the correlation among FF switching activity, the smaller the required number of features. Hence, the number of key FFs does not necessarily increase with circuit size, but is rather dependent on functionality.

Next, the table reports the $P_{dyn}$ estimation error of the original floating point regression model, and the one of the corresponding fixed-point hardware meters (columns labeled **Float Error** and **Fixed Error** respectively). Each column reports the $MAE$, as well as the error obtained adding twice the standard deviation $\sigma$ to the mean (i.e. $MAE + 2\sigma$). Under normality assumptions, the latter can be interpreted as a 95% confidence interval. Taking the FIR software-based meter as an example, this means that the error will be smaller than 6.52% for 95% of the workloads. These values are comparable with the $P_{leak}$ MAE obtained for the ring oscillator with the aforementioned parameters, i.e. 2.32%, with a 95% confidence interval of 5.54%. Notice that the final fixed-point error only slightly increases with respect to the original floating-point model. The former is computed after the final retrain phase, hence it also accounts for the modifications in the original circuit caused by the insertion of the meter itself.

TABLE I
$P_{dyn}$ ERRORS AND METER OVERHEADS FOR THE THREE BENCHMARK CIRCUITS. TARGET ERROR THRESHOLD: $MAE = 3\%$.

| Circuit | $N_{FF}$ | Type | M | Float Error [%] MAE (MAE + $2\sigma$) | Fixed Error [%] MAE (MAE + $2\sigma$) | Fast-fast Error MAE (MAE + $2\sigma$) | Area Ovr. [%] | Power Ovr. On [%] | Power Ovr. Off [%] |
|---|---|---|---|---|---|---|---|---|---|
| **FIR** | 10 | **SW** | 9 | 2.55 (6.52) | 2.59 (6.59) | 2.93 (7.47) | 9.74 | 12.43 | 1.79 |
| | | **HW** | 11 | | 2.57 (6.53) | 2.91 (7.54) | 31.69 | 22.08 | 5.21 |
| **AES** | 4 | **SW** | 9 | 2.74 (6.80) | 2.79 (6.31) | 2.89 (7.44) | 0.49 | 0.50 | 0.13 |
| | | **HW** | 10 | | 2.78 (6.12) | 2.87 (7.39) | 1.79 | 1.63 | 0.71 |
| **RI5CY** | 8 | **SW** | 10 | 2.43 (6.27) | 2.44 (6.38) | 2.66 (7.54) | 1.45 | 1.80 | 1.00 |
| | | **HW** | 13 | | 2.52 (6.48) | 2.59 (7.39) | 7.14 | 5.77 | 1.28 |

The last three columns of Table I report area overheads, as well as power overheads during estimation (**Power Ovr. On**) and when the meter hardware is idle (**Power Ovr. Off**). The area overhead is always smaller than 10%, except in the case of the hardware-based meter for the FIR. This is due to the much smaller size of the FIR compared to the other designs. However, a FIR is unlikely a standalone design, and is normally part of a larger signal processing SoC. Thus, the software-based estimator is the most interesting alternative for this design. For the AES and RI5CY, instead, both solutions are equally interesting, and represent a trade-off between circuit size and the involvement of software.

Power overheads must be read considering that the meter will exhibit a very small duty cycle. For example, at $V_{dd} = 0.9\,\mathrm{V}$, the maximum clock frequency for the RI5CY core is $f_{clk} \approx 800\,\mathrm{MHz}$. At this frequency, the software-based meter takes about $1.38\,\mu\mathrm{s}$ to complete a full estimation. Considering that few tens of µs are necessary just to complete a $(V_{dd}, f_{clk})$ transition in a DVFS system [10], the minimum time interval for power management in a core is in the order of $100\,\mu\mathrm{s}$ to $1\,\mathrm{ms}$. This means that the meter will be in idle state most of the time, and although the power overhead during estimation is not negligible, the total *energy* overhead is definitely minimal.

### C. Impact of variability

All previous experiments were executed considering a slow-slow process corner. As a final experiment, we tested the performance of our meters in response to variability in the manufacturing process. To do so, we performed a new final retrain of the models using power data collected from analyses in fast-fast corner. Error results for $P_{dyn}$ are shown in the column labeled **Fast-fast Error** of Table I. For $P_{leak}$, the MAE in fast-fast becomes 2.73%, with a 95% confidence interval of 7.55%. As shown, the MAE increases slightly, but still remains below the user-imposed threshold of 3%. This experiment only mimics the effect of retraining with measurements on a real chip. However, it gives a confirmation that, although regression coefficients are surely process-dependent, key FFs do not change, and accuracy can be restored after retrain.

## V. CONCLUSIONS

We have presented a novel method for estimating dynamic and static power in generic circuits, via the insertion of fully-digital switching activity meters in key locations. We have shown how these meters can be used to estimate $P_{dyn}$, either in software or fully in hardware. Moreover, we have shown that a similar regression approach, applied to operating conditions, can be used for estimating $P_{leak}$ as well. The accuracy of such models is comparable to that obtained using performance counters, with the advantage of having an estimate of the power breakdown, and not being restricted to core-based systems. Future developments of this work include extensions of the models to secondary variables for greater accuracy, and the usage of the proposed meters for building advanced power management policies.

## REFERENCES


[1] S. M. Martin et al, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in *ICCAD 2002.*, pp. 721–725.

[2] M. Cochet et al, "Experimental model of adaptive body biasing for energy efficiency in 28nm utbb fd-soi," in *S3S Conf. 2014*, pp. 1–2.

[3] D. Puschini et al, "Body bias usage in utbb fdsoi designs: A parametric exploration approach," *Solid-State Electron.*, vol. 117, pp. 138–145, 2016.

[4] D. J. Pagliari et al, "A methodology for the design of dynamic accuracy operators by runtime back bias," in *DATE 2017*, pp. 1165–1170.

[5] E. Beigne et al, "A 460 MHz at 397 mV, 2.6 GHz at 1.3 V, 32 bits VLIW DSP Embedding F MAX Tracking," in *IEEE JSSC*, vol. 50, no. 1, pp. 125-136, Dec. 2014.

[6] P.-E. Gaillardon et al, "A survey on low-power techniques with emerging technologies: From devices to systems," *J. Emerg. Technol. Comput. Syst.*, vol. 12, no. 2, pp. 12:1–12:26, Sep. 2015.

[7] J. Rabaey, *Low Power Design Essentials*, 1st ed. Springer, 2009.

[8] V. Venkatachalam and M. Franz, "Power reduction techniques for microprocessor systems," *ACM Comput. Surv.*, vol. 37, no. 3, pp. 195–237, Sep. 2005.

[9] T. Kolpe et al, "Enabling improved power management in multicore processors through clustered dvfs," in *DATE 2011*, pp. 1–6.

[10] S. Park et al, "Accurate modeling of the delay and energy overhead of dynamic voltage and frequency scaling in modern microprocessors," *IEEE TCAD*, vol. 32, no. 5, pp. 695–708, May 2013.

[11] C. Kim and K. Roy, "Dynamic vth scaling scheme for active leakage power reduction," in *DATE 2002*, pp. 163–

[12] E. Beigne et al, "Ultra-wide voltage range designs in fully-depleted silicon-on-insulator fets," in *DATE 2013*, pp. 613–618.

[13] X. Jiang et al, "Micro power meter for energy monitoring of wireless sensor networks at scale," in *IPSN 2007*, pp. 186–195.

[14] K. Singh et al, "Real time power estimation and thread scheduling via performance counters," *SIGARCH Comput. Archit. News*, vol. 37, no. 2, pp. 46–55, Jul. 2009.

[15] W. L. Bircher and L. K. John, "Complete system power estimation using processor performance events," *IEEE T. Comp.*, vol. 61, no. 4, pp. 563–577, April 2012.

[16] R. Rodrigues et al, "A study on the use of performance counters to estimate power in microprocessors," *IEEE TCAS II*, vol. 60, no. 12, pp. 882–886, Dec 2013.

[17] H. Nagasaka et al, "Statistical power modeling of gpu kernels using performance counters," in *ICGC 2010*, pp. 115–122.

[18] V. Peluso et al, "Ultra-fine grain vdd-hopping for energy-efficient multi-processor socs," in *VLSI-SoC 2016*, pp. 1–6.

[19] P. K. Chundi et al, "Hotspot monitoring and temperature estimation with miniature on-chip temperature sensors," in *ISLPED 2017*, pp. 1–6.

[20] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[21] "Pulp: An open parallel ultra-low-power processing-platform," http://www.pulp-platform.org/.