Leakage Minimization Using Self Sensing and Thermal Management

Alireza Vahdatpour
Computer Science Department
University of California, Los Angeles
alireza@cs.ucla.edu

Miodrag Potkonjak
Computer Science Department
University of California, Los Angeles
miodrag@cs.ucla.edu

ABSTRACT

We have developed a system architecture, measuring and modeling techniques, and algorithms for on-line power and energy optimization and thermal management. The starting point for our approach is a simple and small gate-level network that can be used for real-time and low overhead measurement of temperature on chip positions where our network gates are placed. We use linear programming and interpolation to calculate the temperature at any arbitrary point of the integrated circuit. The periodic calculations of the temperature are used to estimate locally dissipated energies, which are consequently used to derive the most efficient use of operational times to minimize the overall leakage energy. All concepts and algorithms are experimentally validated using a simulation platform that consists of the Alpha 21364 processor and the SPEC benchmarks.

Categories and Subject Descriptors

B.7.m [**Hardware**]: Integrated Circuits—*Miscellaneous*

General Terms

Measurement, Design

Keywords

Leakage Energy, Thermal Management, Delay

1. INTRODUCTION

There are two principal forces that created an impetus for our research: (i) need for accurate, low latency, high spatial and time resolution dynamic thermal monitoring and run-time management in order to optimally use the available energy while satisfying all timing constraints; and (ii) importance of accurate and dynamic, yet low cost and low overhead monitoring of integrated circuits (ICs) using their own circuitry with respect to a variety of design metrics such as substrate and crosstalk noise, temperature and energy, and aging and pending failures.

Thermal management and techniques for reducing leakage power and energy in ICs have emerged as a popular research topic. While leakage power percentage in current technologies is still relatively low, it is bound to significantly increase with each new IC generation. In this paper we restrict our focus on thermal self-measurements and thermal run-time management. The special emphasize is placed on minimizing the consumed leakage energy over time. We introduce several conceptual and technical innovations for thermal and power monitoring and management. Maybe the best way to summarize them and their role is to briefly explain the new overall approach, which consists of the following four phases: **(i)** Self-sensing network. We create and superimpose a simple gate network over the actual design of the IC. The self-sensing network and the design are completely disjoint in order to enable their simultaneous operation. The self-sensing network imposes no performance overhead. Also, power and energy overheads are negligible since the size of the network is much smaller and the network is much less frequently used. By using the available white spaces, the hardware overhead can be essentially eliminated or greatly minimized. Since the delay is proportional to the temperature of gates, we can obtain temperature of locations close to the network gates, by measuring the delay of the gates in the self-sensing network. **(ii)** Numerical and algebraic thermal interpolation. Once we measured the delay of the circuit, we use linear programming formulation and interpolation techniques to calculate the temperature at any arbitrary point of interest at the chip. **(iii)** Energy-centric approach. We use two or more consecutive temperature measurements to calculate the electrical energy transformed into the thermal energy at each filed of a grid imposed over the IC. The key observation is that while the current temperature depends on the long sequence of previous temperatures, the energy information is independent from historical data. **(iv)** Real-time measurement-based temperature and energy management. The locally generated heat energy depends on the number and characteristic of local gates and the input data used by the functionality. It is well known that the same program may consume often several times and sometimes even several orders of magnitude more or less energy and execution time depending on the input data. We have developed a real-time methodology to interleave cooling and operational intervals, depending on the runtime thermal activity in such a way that leakage energy is minimized. The rest of the paper is organized as of the following: Section 2 covers related work and preliminaries. In Section 3 we introduce our self sensing architecture and present its evaluation. Section 4 presents the leakage minimization technique and Section 5 contains

ISLPED'10, August 18–20, 2010, Austin, Texas, USA.

the simulation results. Finally, the paper is concluded is Section 6.

2. RELATED WORK AND BACKGROUND

In the last two decades, energy emerged as a premier design and run-time management metric. Thermal management has attracted a great deal of attention and a variety of low power techniques have been proposed [1, 2, 3]. There are recent studies (i.e. [17]) on using digital design blocks for temperature measurement. The measurement circuit in such studies are generally large, and therefore, temperature readings are average values for wide areas in an IC. In addition, approaches by [9] and [10] use processor specific features such as *performance counters* to estimate processor temperature in the software during the runtime, which is applicable to few processor families supporting the feature and leads to high performance overhead in execution time. Recently, Potkonjak et. al [7] have shown how the post silicon gate-level characterization (in presence of manufacturing variability) is viable using input vector control techniques. We use similar methodology to measure the delay of the sensor gates. While [6] has presented a temperature management technique for leakage minimization, our approach has fundamental differences. [6] assumes the thermal behaviors of the task and processor are known a priori. In addition, it assumes that the leakage energy consumption of the processor is negligible in sleep mode, which requires extra power gating and supply control logic to be implemented in the processor. In addition, up to our knowledge, none of the studies in leakage management considered the spatial variation of the processor temperature.

2.1 Delay Model and Measurement

The impact of temperature on delay has been studied widely. Depending on the fabrication process and the circuit technology, several timing-temperature models have been proposed (e.g. [12, 11]). In this study, we used the linear temperature-delay model introduced by [11]. According to this study, the variation of the delay is about 6%, when temperature varies by 50°C (starting from 50°C).

We use the scheme from Figure 1.a to measure the delay of a logic circuit. Clock cycle period can be dynamically changed in a circuit by the resolution of pico seconds [13]. At the same time, considering gigahertz and megahertz clock frequencies of integrated circuits (where the clock period is larger than hundreds of pico seconds), high resolution dynamic management of clock frequency is feasible. To accurately measure the propagation delay of a circuit, it is enough to continuously and slightly increase the clock frequency of the *measurement circuit* (see Figure 1), until the clock cycle period becomes less than the propagation delay of the circuit. Upon reaching this point, the change in the output of the circuit (*Output*) will not affect FF_1 and *Out* will be different than *Output*. This clock cycle period can be assumed to be the propagation delay of the logic circuit. Figures 1.b and 1.c depict the conditions where the clock period is slightly more and less than the propagation delay (t_0) of the combinational circuit of Figure 1.a. Note that in this example, the *Output* of the combinational circuit is assumed to become high as the input becomes high.

2.2 Leakage Energy Model

Energy consumption of an IC consists of dynamic and

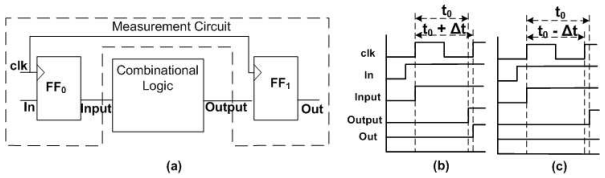

Figure 1: a) A simple mechanism for delay measurement b-c) Circuit behavior when propagation delay is smaller and greater than the clock period

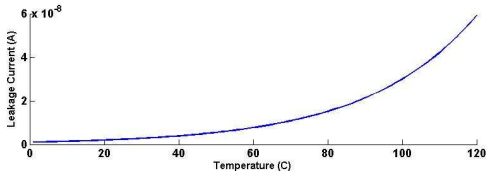

Figure 2: Leakage current vs. temperature

leakage energies. In CMOS technology, dynamic energy consumption directly depends on the working voltage level and the switching activity of transistors. Leakage power is resulted by the flow of leakage current through transistors and is modeled to be:

$$P_{leakage} = N \cdot I_{leakage} \cdot V_{dd}$$

where N is the number of transistors in the circuit and $I_{leakage}$ is a function of several parameters, most importantly the thermal voltage (v_t), supply voltage (V_{dd}), and threshold voltage (v_{th})[5]:

$$I_{leakage} = A \cdot e^{\alpha V_{dd} + \beta} \cdot v_t^2 \cdot (1 - e^{\frac{-V_{dd}}{v_t}}) \cdot e^{\frac{\gamma v_{th} + \delta}{v_t}}$$

where both v_t and v_{th} are functions of temperature and A, α, β, γ, and δ are constant coefficients which depend on transistor technology and fabrication process. Figure 2 depicts how leakage current of a nominal transistor increases significantly by increasing the temperature. As an illustrating example, [4] reports that while at $30°C$ leakage power is only 6% of the total power consumption, it becomes 56% of the total power at $110°C$.

3. SELF SENSING NETWORK

The most basic approach for temperature sensing is to place thermal sensors in the IC package. This solution has clear shortcomings. Not only are thermal sensors (diodes) costly, their size is relatively large. Additionally, the rate at which the die temperature can change is increasing to the point that the currently available thermal sensor interface logic is too slow to allow reliable die temperature measurement or closed loop thermal control [8]. Recent approaches by [9] and [10] use processor specific features such as *performance counters* to estimate IC temperature at runtime. In addition to being inaccurate ([10] reports 10% error), the required software mechanism adds about 4 to $8°C$ of thermal overhead and can slow down the execution of applications by up to 54% [9]. The major disadvantage of current temperature monitoring mechanisms is in the spatial granularity of the sensing. Generally, these methods account the dynamic energy consumed by the processor or its main functional blocks to estimate the average temperature of the processor die or its functional blocks. We will show that due to the high spatial variation of the temperature, using such mechanisms for power management results in high inaccuracy. Our technique requires placement of a low overhead

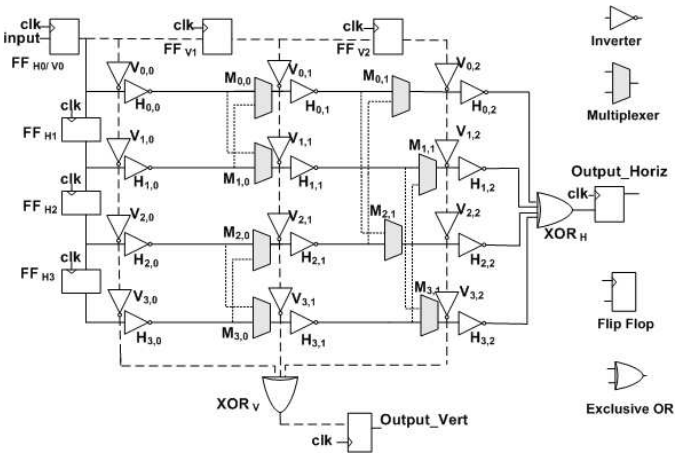

Figure 3: A 4x3 sensory circuit

sensory circuit in the IC. The self-sensing circuit consists of basic logic gates with controlled manufacturing characteristics (e.g. gate size). We have designed a circuit with minimum number of dedicated inputs and outputs, so that it can be easily integrated with any IC. By monitoring the delay introduced from temperature variation on each of the gates in the sensory circuit, we estimate the temperature using the relation between the temperature and the delay of gates.

3.1 Sensory Circuit Architecture

Figure 3 depicts a sample structure of the proposed temperature monitoring circuit. In general, this network consists of m rows and n columns of inverters; Each row consists of n inverters and $n-1$ multiplexers and each column is a chain of m inverters. The multiplexer placement is such that 2^{n-1} combinations of inverters chains can be made on each row by changing the select inputs of the multiplexers (connecting inverters from different rows to each other). As a result, in addition to the n vertical inverter chains, $m \cdot 2^{n-1}$ horizontal inverter chains can be constructed dynamically in the network.

The depicted network has only one input and two output ports. Flip-flops $FF_{H0}..FF_{Hm-1}$ and $FF_{V0}..FF_{Vn-1}$ are placed on the input side of the cascaded inverters, and the other end of the inverter chains are connected to exclusive-or gates. The chain of flip-flops results in activation of a single horizontal and a single vertical inverter chain in each clock cycle (fed with the new value of the input, considering the input of the circuit is toggled every $max(m,n)$ cycles) and cause the change in the value of *Output_Horiz* and *Output_Vert*. Note that the output of the exclusive-or gate will change anytime one of its inputs changes. As described in Section 2.1, by changing the clock frequency of the flip-flops on both sides of the inverter chains, the propagation delay can be measured. Upon measuring the propagation delay of the inverter chains, we use a linear programming approach to calculate each multiplexer and inverter delay from the measured chain delays. Briefly describing the linear program, in each equation the measured delay is equal to the sum of the delays of the gates on the active inverter path plus the delay of the XOR gate. Here we omit the full description of the linear program due to space limitation. Interested readers may contact the authors for extended description.

Upon solving the linear equation, estimated delays for each inverter are used to estimate the temperature of the

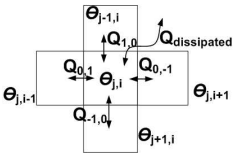

Figure 4: The heat conduction and convection between cells in a silicon grid and the environment

IC at the sensory circuit gate locations. We use the model introduced in Section 2.1 to calculate the temperature variation using propagation delay values. Thereafter, a two-dimensional linear interpolation is used to extend the temperature estimation for the whole IC surface.

3.2 Activity and Thermal Energy Measurement

Beside introducing the approach of using gate delays for fine grain temperature monitoring, we propose a method to monitor the energy consumption of the IC with high spatial and temporal resolution. In addition to the current activity of the IC, temperature of the IC also depends on the previous activity of it and also heat conduction and convection. Hence, it cannot be used as a history independent metric for evaluating processor activity. In the following, using the temperature information, we derive accurate values of energy consumption with high spatial and temporal resolutions (energy consumption and thermal energy terms are used interchangeably, since consumed energy is dissipated in forms of heat). To do so, first we divide the surface of the IC into a grid. Each cell in the grid has a temperature sensor gate placed on it, using the sensing circuit described in the previous section. Figure 4 depicts five neighboring cells of a grid. Lets assume that $\theta_{j,i}$ and ${\theta'}_{j,i}$ are the temperature values of the grid cell located at row j and column i at time t and $t+\Delta t$. According to the heat convection and conduction laws, and assuming that time period Δt is reasonably small, such that variation of temperature is linear in that time, the following equations are valid for the cell j,i:

$$\Delta Q_{dissipated} = H \cdot A_{surface} \cdot (\frac{\theta_{j,i} + {\theta'}_{j,i}}{2} - \theta_{air}) \cdot \Delta t$$

$$\Delta Q_{absorbed} = m \cdot c \cdot (\theta_{j,i} - {\theta'}_{j,i}) \cdot \Delta t$$

$$\Delta Q_{0,\pm1} = -k \cdot A_{side} \cdot \frac{\frac{(\theta_{j,i}+{\theta'}_{j,i})}{2} - \frac{(\theta_{j,i\pm1}+{\theta'}_{j,i\pm1})}{2}}{\Delta x} \cdot \Delta t$$

$$\Delta Q_{\pm1,0} = -k \cdot A_{side} \cdot \frac{\frac{(\theta_{j,i}+{\theta'}_{j,i})}{2} - \frac{(\theta_{j\pm1,i}+{\theta'}_{j\pm1,i})}{2}}{\Delta x} \cdot \Delta t$$

Where H and k are the heat transfer and the silicon conductivity coefficients. m is the mass of the silicon in the cell, and c is the specific heat coefficient for silicon. A_{side} and $A_{surface}$ are the thickness and the area of the grid cells. Also, $\Delta Q_{dissipated}$, $\Delta Q_{absorbed}$, $\Delta Q_{0,\pm1}$, and $Q_{\pm1,0}$ are the thermal energies dissipated through the air, absorbed in the silicon in forms of heat, and transferred between neighboring cells via heat conduction. In addition, according to conservation of energy law, the total amount of consumed and dissipated energies are equal:

$$\Delta Q_{consumed} = \Delta Q_{absorbed} + \Delta Q_{dissipated} + \Delta Q_{0,\pm1} + \Delta Q_{\pm1,0}$$

where $\Delta Q_{consumed}$ is the total energy consumed by the logic gates inside the cell during Δt. As discussed, $\Delta Q_{consumed}$ is an independent metric for each cell, and can be used as the criteria to locate units that are consuming substantial energy in the IC.

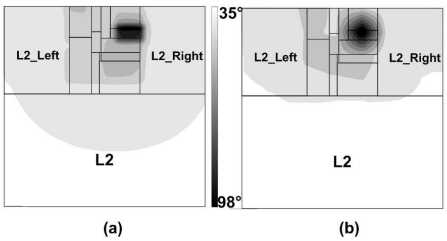

Figure 5: Temperature map generated from iterative execution of *facerec* on EV7: (a) High resolution simulation (b) Estimation by an 8x8 sensor architecture

Table 1: The effect of changing the sensing architecture size on the accuracy of temperature estimation

size	ammp	applu	aspi	art	bzip2	equake	facerec
16^2	0.17	0.14	0.24	0.14	0.26	0.11	0.44
8^2	0.16	0.14	0.21	0.18	0.31	0.11	0.35
4^2	0.49	0.43	0.76	0.41	0.84	0.35	1.25
2^2	0.64	0.55	0.98	0.62	1.08	0.45	1.52

3.3 Analysis of The Sensing Architecture

We have used HotSpot [15] to profile the execution of the SPEC 2000 benchmark suite [14] applications on a 90-nm technology Alpha 21364 (EV7) processor. To evaluate our self-sensing network characteristics, we fed the temperature maps (matrix of size 64×64) extracted from HotSpot into sensing architectures with different sizes, using the reverse of the temperature delay model. Upon estimating the temperature using the sensing architecture and linear programming, we performed linear interpolation to derive a 64×64 matrix of temperature values for the IC. Figures 5.a and 5.b depict two temperature maps, generated by Hotspot and an 8×8 sensor architecture.

To examine the impact of the size of the sensing architecture on the accuracy of temperature estimation, we performed temperature sensing with architectures of size 2^2, 4^2, 8^2, and 16^2. Table 1 depicts the L2 error norm of estimating the IC temperature, comparing to results that HotSpot generated. Increasing the size of the sensing architecture results in lowering the estimation error.

It is worth to mention that delay also depends on other parameters such as manufacturing variability (MV) and supply voltage (SV). Sensitivity to these parameters increases with each new technology node. The MV impact can be eliminated using an additional one time measurements that are conducted after the manufacturing. In these measurements, the nominal delays of the gates at a controlled temperature are measured, and stored to a lookup-table, to normalize the runtime temperature calculations. We used delay as a metric for temperature, since its measurement is fast and cheap. To guarantee the accuracy of temperature estimation using delay (also dependent on SV variation), multi-modal measurements can be utilized. For example, by periodical measurements of leakage and switching power, the estimations are validated.

4. LEAKAGE MINIMIZATION

In this section, we study the temperature behavior of processors and present an execution time mechanism to minimize leakage energy consumption based on the proposed temperature self-sensing network. The dashed line in Figure 6 depicts the thermal behavior of the Alpha processor,

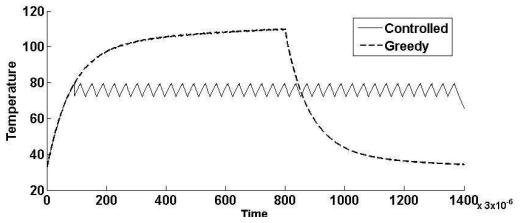

Figure 6: Maximum temperature for executing *facerec* in greedy and controlled modes

when executing several iterations of a benchmark task (*facerec*). Processor's temperature increases exponentially, until it reaches its thermal equilibrium point. The temperature remains in the equilibrium point until the execution finishes and the processor enters the sleep mode, in which the temperature decreases exponentially. It is clear that executing the task in high temperature will result in high leakage energy consumption. A naive approach to minimize the leakage energy is to add sleep intervals (for cooling down the processor) in the task execution (Figure 6 depicts an example). Since the cool down process becomes significantly slower in lower temperatures, keeping the IC in low temperatures results in the increase in the total time required for the task execution, which may result in higher total leakage energy consumption (processor consumes leakage energy even in the sleep mode). For example, keeping the IC temperature in the range of $46 - 48°C$ for the *facerec* will make the execution time 5 times larger and hence the total dissipated leakage energy becomes 2.5 times larger.

Studies such as [6] suggest dynamic programming approach to optimally allocate execution and sleep times to a task, according to operational temperature. They assume that the total runtime of the task, and the thermal behavior of the processor for the specific task are known a priori. However, as [16] suggests, even runtime of the same task significantly changes based on the variation of the input data. Moreover, other parameters such as the ambient temperature also cause variation in the thermal behavior of the processor. In contrast to previous studies, we define the problem of leakage minimization for running a task to be minimizing the total leakage energy of the processor during both execution (heat up) and sleep (cool down) modes. Up to our knowledge, previous studies have only considered IC energy consumption in active mode, which is not realistic.

4.1 Optimal Temperature for Leakage

it is required to find the optimal thermal operation point, where the trade-off between the increase in the execution time (due to addition of sleep intervals) and the reduction in leakage power (due to reduction of the IC temperature) is optimized. Assuming $[\theta_1 ... \theta_2]$ to be the desired operational temperature range, the power intensity of the task execution being stable, and the initial processor temperature to be θ_0, the total time required to execute a task is:

$$T = T_{w(\theta_0,\theta_1)} + n \cdot T_{w(\theta_1,\theta_2)} + (n-1) \cdot T_{s(\theta_2,\theta_1)}$$

where $n = \frac{T_w - T_{w(\theta_0,\theta_1)}}{T_{w(\theta_1,\theta_2)}}$. T_w is the total workload time (the time for execution without interruption), and $T_{w(\theta,\theta')}$ and $T_{s(\theta,\theta')}$ are the time periods that the IC temperature rises or falls from θ to θ', during active or sleep modes, respectively. n shows how many active and sleep intervals are required to perform the task. Finally, the total leakage energy is:

$$E_{leakage} = E_{w(\theta_0,\theta_1)} + n \cdot E_{w(\theta_1,\theta_2)} + (n-1) \cdot E_{s(\theta_2,\theta_1)}$$

Table 2: Ratio of leakage energy consumption in managed mode to the greedy mode

Benchmark	applu	ammp	bzip2	equake	facerec	mesa
Max temp.	63.9	58.2	84.0	50.6	110	73.7
Optimal temp.	52	42	60	43	72	52
w/o cooldown	1.09	1.19	1.08	1.14	1.23	1.06
with cooldown	0.78	0.82	0.78	0.79	0.67	0.79

Here $E_{w(\theta_0,\theta_1)}$, $E_{w(\theta_1,\theta_2)}$, and $E_{s(\theta_2,\theta_1)}$ represent consumed leakage energy during time intervals $T_{w(\theta_0,\theta_1)}$, $T_{w(\theta_1,\theta_2)}$, and $T_{s(\theta_2,\theta_1)}$ respectively. It should be again noted that in the above calculations, for simplicity we assumed that the power intensity of the task execution is not changing during the runtime. In general case, calculating n requires complete energy profile of the task execution. As it was mentioned, depending on the workload of the task, the thermal equilibrium temperature differs and hence the exponential rate of the temperature rise and fall curves varies. Therefore, based on the above formulation, the optimal temperature range for operation of each task and each set of inputs differs. Assuming that the equilibrium point and T_w are known, an off-line solution can find the best operational range by calculating the total leakage energy consumed by the IC for all operational temperature ranges.

Although the off-line method can determine the optimal temperature range for the operation, the assumption that the thermal behavior and the dynamic of task execution are known is unrealistic, since not only tasks behavior and runtime length change by the variation in data inputs, but also the processors temperature may get affected by other processes, interrupts or variations in the cooling mechanism. Next, we present a real-time heuristic mechanism to determine a beneficial operational temperature range.

4.2 Real-time Mechanism

According to equations in section 4.1, an operational temperature range with lower leakage energy and larger $T_{w(\theta_1,\theta_2)}$ is a better choice for minimizing total leakage energy, since it results in decreasing the value of second and third terms of the equation. Assuming $\theta_2 - \theta_1 = 1^\circ C$, Figure 7 depicts the behavior of $R(\theta_1,\theta_2)$ and $E_{Leakage}$ when the the temperature of the processor is kept in $[\theta_1..\theta_2]$ for the *facerec* benchmark, where R is defined to be:

$$R(\theta_1,\theta_2) = \frac{T_{w(\theta_1,\theta_2)}}{E_{w(\theta_1,\theta_2)} + E_{s(\theta_2,\theta_1)}}$$

While calculating the total dissipated leakage energy requires knowledge of the length of the execution and the thermal behavior of the task, R only depends on the leakage energy consumption of the processor between θ_1 and θ_2 (which we use our sensory circuit to measure), and the information regarding heat dissipation in the corresponding temperature range for cooling down, which can be precisely calculated based on the Newton's law of cooling. Therefore, R can be calculated in real-time to predict the total energy consumption variation. Note that while in the depicted example, R is completely correlated with the total leakage energy, in general case, task workload variation changes exponential rate of temperature variation and hence $T_{(\theta_1,\theta_2)}$ differs. Therefore, R cannot optimally predict the total energy consumed in the runtime.

Algorithm 1 presents our real-time mechanism for minimizing the leakage energy. This procedure is periodically executed to monitor the energy consumption (Q_t) and the temperature (θ_t) of the processor and decides whether the processor should go to sleep mode or not. A peak detection method is used to detect the maximum value of R. We use the average value of R in the last α iterations to avoid selection of local maximum values. Upon detecting the peak value of R, temperature is watched to lie in the operational temperature range by switching the processor mode between sleep and active modes. β determines the granularity of operational temperature range. While a smaller β results in more frequent switching between active and sleep modes, it also requires higher granularity of temperature sensing.

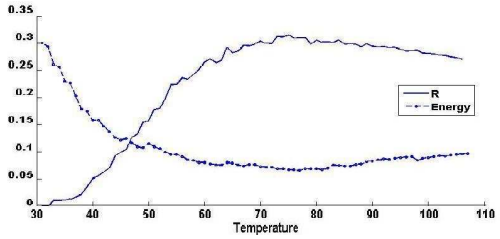

Figure 7: Variation of R and Energy vs. θ_1, where $\theta_2 - \theta_1 = 1$. Values on Y axis are normalized.

Algorithm 1 The real-time method for leakage minimization

```
Input: {θ_0, ..., θ_t}, {Q_0, ..., Q_t}
Output: OperationMode(t) {OperationMode(t) = Active or Sleep}
if bestRange is not assigned yet then
    R(t) ← (θ_t − θ_{t−1}) / (Q_t + Cooling(θ_t, θ_{t−1}))
    if avg(R(t−α), ..., R(t)) > avg(R(t−α−1), ..., R(t−1)) then
        OperationMode(t) ← Active; return
    else
        bestRange ← (θ_t, θ_t − β)
        OperationMode(t) ← Sleep; return
    end if
end if
if θ_t > max(bestRange) then
    OperationMode(t) ← Sleep; return
else if θ_t < min(bestRange) then
    OperationMode(t) ← Active; return
else
    OperationMode(t) ← OperationMode(t − 1); return
end if
```

5. SIMULATION RESULTS

Following the simulation steps discussed in Section 3.3, we evaluated the leakage minimization algorithm using the SPEC2K benchmarks. Unlike other studies that use single temperature to measure the leakage power of the IC, we used the high spatial resolution temperature of the processor to estimate the leakage power. Therefore, assuming the IC surface is divided into an $m \times n$ grid, the total leakage power of the processor is calculated by:

$$P_{leakage} = \sum_{i=1}^{m}\sum_{j=1}^{n} N_{[i,j]} \cdot I_{leakage[i,j]} \cdot V_{dd}$$

where $N_{[i,j]}$ and $I_{leakage[i,j]}$ denote the number of gates and the leakage current in each block of the grid. As mentioned, spatial temperature variation is high on the processor surface, therefore, using one temperature value for calculating leakage energy is a biased observation. While applying temperature controlling mechanisms causes huge leakage energy reduction in hot spot areas, the leakage energy consumption in the cooler area of the IC remains in the same level. Since hot spots usually occupy small portion of the processor, we believe that overall leakage energy saving by means of temperature management cannot be in the range of 30-40%

Table 3: Ratio of leakage energy and execution time for different lengths of operational ranges ($\frac{realtime}{greedy}$)

	applu		ammp		art		bzip2		equake		facerec		galgal		gap		mesa	
	E	T	E	T	E	T	E	T	E	T	E	T	E	T	E	T	E	T
1°	0.95	1.12	0.96	1.21	0.96	1.21	0.94	1.11	0.96	1.15	0.90	1.46	0.93	1.26	0.95	1.10	0.94	1.10
2°	0.96	1.14	0.96	1.28	0.96	1.26	0.94	1.13	0.96	1.22	0.91	1.61	0.94	1.14	0.96	1.14	0.95	1.14
3°	0.96	1.19	0.96	1.34	0.96	1.31	0.94	1.19	0.96	1.27	0.92	1.64	0.94	1.15	0.96	1.19	0.95	1.18
4°	0.96	1.23	0.96	1.37	0.96	1.36	0.95	1.19	0.96	1.32	0.92	1.69	0.95	1.20	0.96	1.23	0.95	1.21

(reported by previous studies) unless the whole processor package becomes significantly hot.

To evaluate the leakage energy consumed by uninterrupted execution of each benchmark (*greedy* approach), we executed each benchmark 40 times in order to let the processor reach its thermal equilibrium, and then we allowed the processor to cool down to reach the ambient temperature. Thereafter, based on the thermal activity model, we calculated the optimal operational temperature range for each benchmark. Table 2 reports the ratio of leakage energy consumed when the Algorithm 1 uses the off-line calculated optimal temperature range as the operational range. We have compared it to the leakage energy consumption of the greedy approach in two scenarios; with and without considering the leakage energy that is consumed when the processor is cooling down after the completion of the task execution. The maximum temperature in the table denotes the temperature of the hottest spot during the execution of the benchmark (as a notion of power intensity of the benchmark).

To evaluate the real-time mechanism for energy saving, we set β to $1°C$, $2°C$, $3°C$, and $4°C$ respectively. In addition, we measured the leakage energy that the greedy approach consumes in the time interval that the real-time mechanisms requires to finish the execution of the task. This way, we compare our algorithm with the greedy mode, with respect to the total time that can be dedicated to the processor to execute a benchmark. As results are reported in Table 4, T denotes the ratio of the time the processor requires to perform the task by applying the real-time mechanism, to T_w. Comparing to the results of Table 3, the ratio of saving is smaller, since not only is the operational temperature range determined by the on-line algorithm but also the energy of the greedy approach is calculated for the same time that online algorithm finishes the task (not a complete cool down). In other words, when the processor is assigned a slightly longer time period (than T_w) to execute a task, it can use thermal management technique to reduce the leakage energy. Clearly, shortening β causes the processor to operate more closely to its optimal temperature point. However, it requires more frequent and accurate temperature sensing. The real-time mechanism has single temperature and energy values as input. However, we discussed earlier that there is high spatial variation in the temperature of the processor.

6. CONCLUSION

We presented a real-time temperature sensing architecture, energy modeling technique and algorithm for temperature and energy management and optimization. By introducing a low overhead mechanism to monitor the temperature of the integrated circuits, we proposed a model to independently monitor the energy consumption of each unit in the IC. In addition, we presented a real-time mechanism to minimize the leakage energy consumed by the processor to run tasks. We showed that in addition to the operational temperature, operational times can also be managed to reduce the overall leakage energy consumption.

7. ACKNOWLEDGMENTS

This research is partially supported by the Center for Domain-Specific Computing (CDSC) funded by the NSF Expedition in Computing Award CCF-0926127.

8. REFERENCES

[1] D Kirovski, M Potkonjak, System-Level Synthesis of Low-Power Hard Real-Time Systems, DAC 1997, pp. 697-702.

[2] F Dabiri, A Vahdatpour, M Potkonjak, M Sarrafzadeh, Energy Minimization for Real-Time Systems with Non-Convex and Discrete Operation Modes, DATE 2009, pp. 1416-1421.

[3] Y Alkabani et al., Input vector control for post-silicon leakage current minimization in the presence of manufacturing variability, DAC 2008, pp. 606-609.

[4] F Fallah, M Pedram, Standby and Active Leakage Current Control and Minimization in CMOS VLSI Circuits, IEICE Trans. on Electronics, 2005.

[5] Y Zhang, D Parikh, K Sankaranarayanan, K Skadron, Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects, University of Virginia, Computer Science Tech Report, 2003.

[6] L Yuan, S Leventhal, G Qu, Temperature-aware leakage minimization technique for real-time systems. ICCAD 2006, pp. 761-764, CA, USA.

[7] M Potkonjak, A Nahapetian, M Nelson, T Massey, Hardware Trojan horse detection using gate-level characterization, DAC 2009, pp. 688-693.

[8] S Gunther, F Binns, DM Carmean and JC Hall, Managing the impact of increasing microprocessor power consumption, Intel Technology Journal, 2001.

[9] KJ Lee, K Skadron, Using Performance Counters for Runtime Temperature Sensing in High-Performance Processors, IPDPS 2005, page 232.1, DC, USA.

[10] A Merkel, F Bellosa, Balancing power consumption in multiprocessor systems, ACM SIGOPS European Conference on Computer Systems, 2006, pp. 403-414.

[11] C de Benito, S Bota, JL Rosselló, J Segura, Temperature impact on multiple-input CMOS gates delay, Proceedings of SPIE, 2007.

[12] BP Das et. al., Voltage and Temperature Scalable Gate Delay and Slew Models Including Intra-Gate Variations, International Conference on VLSI Design, 2008.

[13] J Kalisz, Review of methods for time interval measurements with picosecond resolution, Metrologis, vol. 41, pp. 17-32, 2004.

[14] SPEC-CPU2000. Standard Performance Evaluation Council, Performance Evaluation in the New Millennium, Version 1.1.

[15] S Ghosh, S Velusamy, K Sankaranarayanan, K Skadron, HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design, IEEE Trans. VLSI Systems, 14(5): 501-513, 2006.

[16] K Gururaj, J Cong, Energy Efficient Multiprocessor Task Scheduling under Input-dependent Variation, DATE 2009, pp. 411-416, France.

[17] P Chen et. al., A Fully Digital Time-Domain Smart Temperature Sensor Realized With 140 FPGA Logic Elements, IEEE Transactions on Circuits and Systems, 54(12): 2661-2668, 2007.