A Low-Voltage Processor for Sensing Applications With Picowatt Standby Mode

Scott Hanson, *Student Member, IEEE*, Mingoo Seok, *Student Member, IEEE*, Yu-Shiang Lin, *Member, IEEE*, ZhiYoong Foo, Daeyeon Kim, *Student Member, IEEE*, Yoonmyung Lee, *Student Member, IEEE*, Nurrachman Liu, *Student Member, IEEE*, Dennis Sylvester, *Senior Member, IEEE*, and David Blaauw, *Senior Member, IEEE*

***Abstract*—Recent progress in ultra-low-power circuit design is creating new opportunities for cubic millimeter computing. Robust low-voltage operation has reduced active mode power consumption considerably, but standby mode power consumption has received relatively little attention from low-voltage designers. In this work, we describe a low-voltage processor called the Phoenix Processor that has been designed at the device, circuit, and architecture levels to minimize standby power. A test chip has been implemented in a carefully selected 0.18 μm process in an area of only 915 $\times$ 915 μm^2. Measurements show that Phoenix consumes 35.4 pW in standby mode and 226 nW in active mode.**

***Index Terms*—Low voltage, ultra-low leakage, ultra-low power.**

I. Introduction

THE prevalence of mobile computing has helped define a vision of complex computational resources in miniscule volumes [1], [2]. As the volumes of computing resources approach one cubic millimeter, active monitoring and actuation can be used to enrich a wide range of applications. Cubic millimeter computing will be particularly important in implantable medical devices, where reducing device volume helps minimize implant damage to the body. The diagnosis and treatment of Glaucoma, for example, requires periodic measurements of pressure in the eye (intra-ocular pressure). Intra-ocular pressure is currently monitored directly by a doctor, requiring frequent trips to the doctor's office to ensure sufficient temporal resolution [3]. An intra-ocular pressure sensor with a MEMS pressure sensor, microprocessor, memory, radio and power source small enough to be implanted in the eye would reduce both cost and time investment and would increase the temporal resolution of pressure measurements.

Although MEMS and circuit components easily meet the volume constraints of intra-ocular pressure sensing and other cubic millimeter computing applications, batteries and energy scavenging power sources cannot be easily miniaturized while also serving the power demands of the MEMS and circuit components. Minimizing the power demands of each component is therefore one of the central challenges in designing a cubic millimeter computing system. Consider a system with a thin film zinc/silver oxide battery with a capacity of 100 μAh/cm^2 and output voltage of 1.55 V [4]. If the battery size is restricted to 1 mm^2, the average system current must be only 114 pA (for power consumption of 177 pW) to guarantee one year of battery life.

For the remainder of this work, we explore power minimization in digital components including the microprocessor, memory, watchdog timers and simple sensors. A growing body of work has studied the use of ultra-low-voltage operation to minimize active mode power consumption in digital circuits. Early work proved that operation at extremely low voltage was possible [8], and later work showed that memory can be redesigned to operate robustly at low voltage [13]–[15]. The authors of [5], [7], [21] demonstrated that unprecedented energy efficiency can be achieved in general purpose subthreshold processors. A number of well-known techniques have also been used widely to reduce standby power including the use of high-V_{th} devices, clock gating, and power gating. As an alternative to power gating, the ultra-low-power processor presented in [23] operated at a lower voltage during standby. The subthreshold processor in [21] applied a similar technique to SRAM during standby but implemented clock gating and power gating on logic as well to achieve standby mode power under 1 μW.

We have developed an ultra-low-energy sensor processor called the Phoenix Processor that leverages low-voltage operation and several well-known standby mode techniques. Unlike prior work, we choose standby power as the primary design metric. We have implemented a much more aggressive standby strategy than past low-voltage processors [5], [7], [21]. This is of particular importance in a typical wireless monitoring system that spends the majority of its lifetime in standby mode. For example, a temperature logger might take sensor measurements once every 10 minutes. Assuming a ~ 100 ms active period and a low-voltage processor that consumes 5$\times$ more power in active mode than standby mode [5], the temperature logger consumes 1200$\times$ more energy in standby mode than active mode. To address this discrepancy in Phoenix, we have implemented an aggressive standby mode strategy including the deliberate selection of an older low-leakage technology, an alternative power gating approach, a custom leakage-optimized instruction set, simple data memory compression, and a new ultra-low-leakage memory cell. In implementing this strategy,

Manuscript received August 25, 2008; revised November 28, 2008. Current version published March 25, 2009.

S. Hanson, M. Seok, Z. Foo, D. Kim, Y. Lee, N. Liu, D. Sylvester, and D. Blaauw are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: hansons@umich.edu).

Y.-S. Lin was with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA. He is now with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2009.2014205

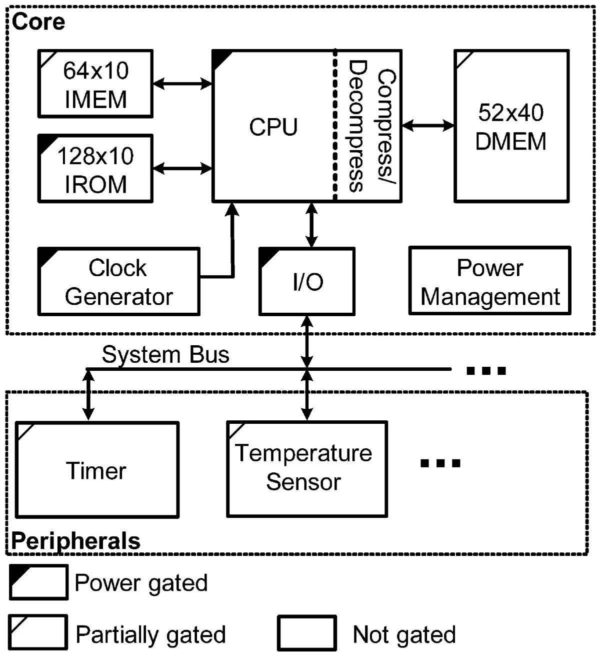


Fig. 1. The Phoenix Processor.

we have dramatically reduced standby power at the expense of area and active energy. These trade-offs will be quantified throughout this work. Measurements of a 0.18 μm test chip reveal that Phoenix consumes 226 nW in active mode and only 35.4 pW in sleep mode. We discuss the Phoenix Processor in detail for the remainder of this paper. We begin in Section II with an overview of the Phoenix Processor components and typical system operation. We discuss accommodations made for standby mode in detail in Sections III–VII. Finally, the 0.18 μm test chip and measured results are discussed in Sections IX–X.

II. System Overview

As shown in Fig. 1, the Phoenix Processor is a modular system with a core unit consisting of an 8-bit CPU, a 52×40-bit data RAM (DMEM), a 64×10-bit instruction RAM (IMEM), a 64×10-bit instruction ROM (IROM) and a power management unit (PMU). The core serves as a parent to peripheral devices, including a watchdog timer and a temperature sensor. The core and peripheral devices communicate over a system bus using a simple asynchronous protocol. The I/O controller addresses up to eight peripherals on the system bus for sensing systems requiring additional peripherals.

In typical operating conditions, the Phoenix Processor spends an extended period of time in standby mode (e.g., 10 minutes) and wakes up in response to an exception raised by the watchdog timer (a 0.9 pW current-starved oscillator). Once awake, the Phoenix Processor polls the temperature sensor and runs a short routine to process and store the measurement. After completing the data processing routine, Phoenix returns to standby mode.

The power consumption in active mode is dominated by components with high switching activity, such as the CPU. To minimize this source of power consumption we scale voltage aggressively to 0.5 V, a sub-threshold voltage (for high-V_{th} devices) or near-threshold voltage (for medium-V_{th} devices) in the target technology. The challenges of low-voltage digital design have been covered extensively in recent literature [5], [7], [8] and will not be the focus of this work. Instead, we place emphasis on accommodations made for standby mode operation. The Phoenix Processor was designed at the device, circuit and architecture levels with the primary goal of standby power minimization. In subsequent sections, we discuss each of the key components of this comprehensive standby mode strategy.

III. Technology Selection

Despite its importance to both power and performance, there has been little investigation of technology selection for low-voltage circuits. The requirements of sub-threshold and near-threshold circuits are different from those of normal super-threshold circuits, and the optimal technology is therefore different. The required performance is much relaxed in typical low-voltage sensing applications, so older technologies can easily meet performance requirements. Furthermore, the long standby time observed in many sensor applications makes cumulative standby leakage energy significant, as we observed earlier in this work. Advanced technology nodes have also been optimized exclusively for super-threshold operation, resulting in sub-optimal noise margins, power and performance [20].

The ideal technology would simultaneously offer small feature sizes and devices with ultra-low leakage. Since no such technology was available for use in academic research, we investigated standard CMOS technologies from 0.25 μm to 65 nm. Newer technologies tend to offer devices with higher subthreshold leakage but smaller capacitance. Conversely, older technologies offer lower subthreshold leakage but larger capacitance, effectively offering reduced standby power at the expense of active power. A simple analysis using the method in [19] with simple inverter-chain models of a CPU and memory and a duty cycle of 0.001 (1 s of active time per 1000 s of standby time) reveals that an older 0.18 μm technology gives optimal energy. The active power penalty paid for adopting an older technology is easily outweighed by the dramatic reduction in standby power. Note that the 0.18 μm technology under study offers a device with a particularly high V_{th}, and further reverse scaling may have been warranted if the 0.25 μm technology offered a similar device. A processor implemented in the energy-optimal 0.18 μm technology is 7.7$\times$ larger than a similar processor in a 65 nm technology, but our analysis reveals that total energy is reduced by 647$\times$. This is an extremely favorable trade-off, especially when the volume of a wireless sensor is dominated by the battery volume (and not die volume).

The selected 0.18 μm technology includes a thin-oxide medium-V_{th} device with V_{th} $\sim$0.5 V and a thick-oxide IO device with V_{th} $\sim$0.7 V. All retentive gates (i.e., those gates that remain awake in standby mode) are implemented using the high-V_{th} devices, which consume $\sim$1000$\times$ less leakage power per unit of gate width than the medium-V_{th} devices. Note that we do not use high-V_{th} devices in non-retentive gates since the minimum dimension is larger than that of the thin-oxide device, which gives both area and active energy penalties. In addition to the selection of an older technology, stack-forcing is used to reduce leakage power further. Leakage reduction due to the stack effect has been shown in previous work to be effective [17]. In our selected technology, stacking two transistors gives $\sim$2$\times$ leakage reduction.

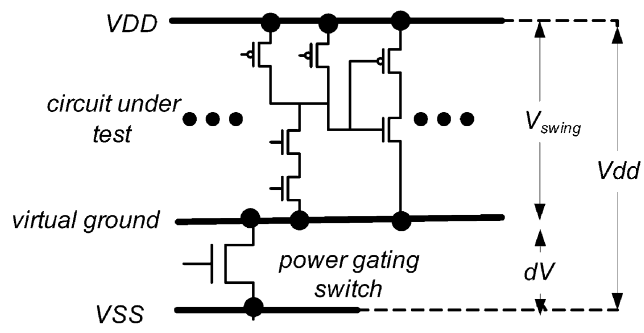


Fig. 2. A typical power gating switch.

IV. Power Gating Under Relaxed Performance Constraints

A power gating switch, as shown in Fig. 2, is often used in low-power circuits to minimize leakage in non-retentive circuit blocks during standby modes. At normal super-threshold operating voltages (e.g., >1 V in 45 nm and 65 nm designs), a high-V_{th} device is typically used as a power gating switch since it delivers comparable on-current to the nominal device with exponentially smaller off-current. Additionally, wide power gating switches are typically used to minimize the performance penalty of power gating.

For cubic millimeter computing applications with modest performance requirements, minimizing standby power is the most important goal. In such applications, performance can be sacrificed for lower leakage, which is in stark contrast to the typical approach to power gating. In the Phoenix Processor, we leverage these modest performance requirements with an alternative power gating approach.

Our power gating approach relies on a medium-V_{th} power switch rather than a high-V_{th} switch as in the typical approach. The on-current of the high-V_{th} device is exponentially smaller than that of the medium-V_{th} device at low voltage. Therefore a high-V_{th} device must be sized up $\sim$1000$\times$ as compared to a medium-V_{th} device to meet the current demands of the primary circuit, which is implemented using medium-V_{th} devices. The area overhead as well as the power overhead of charging/discharging such a large switch is avoided by using a medium-V_{th} power switch at no performance or leakage penalty. Note that the difficulties associated with high-V_{th} power switches in low-voltage circuits as well as the utility of switches with lower V_{th} have been noted previously [25].

In addition to using medium-V_{th} power switches, the strength of our power gating switch compared to the circuit under test is smaller than that of the typical power gating approach. A stronger power gating switch minimizes the performance penalty of power gating at the expense of additional leakage during standby mode. Given the modest performance demands for the Phoenix Processor, we choose to reduce standby mode leakage considerably by selecting a very weak power gating switch [16].

In the Phoenix Processor, the medium-V_{th} power switch is only 0.66 μm, which is 0.01% of total effective NFET width and 3$\times$ larger than the minimum width in the target technology. We increase the length from 0.18 μm to 0.50 μm to improve inverse subthreshold slope and consequently increase the on-current to off-current ratio. The 0.66 μm power gating switch is connected to the CPU and several other logic blocks as shown in Fig. 3. Simulations with a model of the CPU indicate that the virtual ground rail bounces by a maximum of $\sim$ 100 mV, which is sufficient to guarantee correct logic operation. The non-retentive parts of IMEM and DMEM, such as decoders and output buffers, are connected to a separate power gating switch since the robustness of low-voltage memory may be compromised by a voltage drop across the power gating switch. The measured energy and performance implications of our proposed power gating strategy will be discussed in Section IX.

V. CPU and Instruction Set Design for Standby Mode

In accordance with the conclusions of previous studies of subthreshold processor architectures [22], we have selected a simple CPU architecture with two-stage pipeline, 8-bit data width, and 10-bit instruction width to reduce active mode power and standby mode power. The instruction set includes support for basic arithmetic computation in typical sensor logging applications. As shown in Fig. 4, the first pipeline stage consists of instruction fetch and decode as well as a scratch memory with an 8-entry register file and 16-entry cache. The second pipeline stage includes a simple ALU, write-back logic, and a memory interface unit that compresses (decompresses) outgoing (incoming) memory traffic. The ALU includes hardware for addition, subtraction, and shifting. The CPU has been designed to minimize energy in both active and standby modes, as shown in the remainder of this section.

Since the computational demands of cubic millimeter computing applications are typically modest, the CPU was simplified to support a minimum set of operations. Such simplicity reduces decode complexity and eliminates unnecessary switching activity, thus reducing active mode power. Furthermore, elimination of complex operations like multiplication eliminates large, leaky circuit blocks. Since leakage energy can be >30% of total energy in active mode for low-voltage circuits [6], the resulting active mode power savings are significant.

Instruction set architecture (ISA) optimization also plays an important role in minimizing power consumption in standby mode. Since the contents of IMEM must be retained in standby mode, it is important to minimize the instruction width. The leakage penalty of instruction memory can alternatively be eliminated by using flash-based memory, but this requires costly processing steps. The custom ISA for the Phoenix Processor was compressed to an instruction width of only 10 bits by selecting a minimum set of 18 instructions (Table I). The benefits of selecting a narrow instruction width will be quantified using measured results in Section IX.

Efficient operand encoding also helps to reduce the instruction width. Stack-based ISAs give very short instructions since the operands and destination register are implicitly assumed to be at the top of the stack. However, this small instruction size comes at the cost of flexibility offered by the typical approach in which operands are selected within the instruction from a set of general purpose registers. To simultaneously achieve encoding

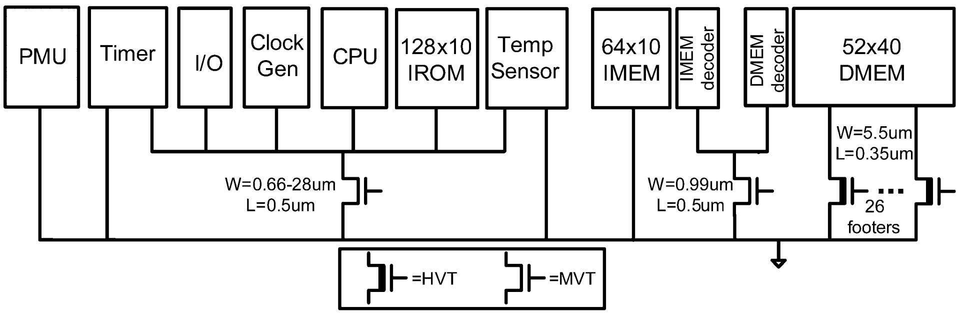


Fig. 3. Footer allocation in the Phoenix Processor.

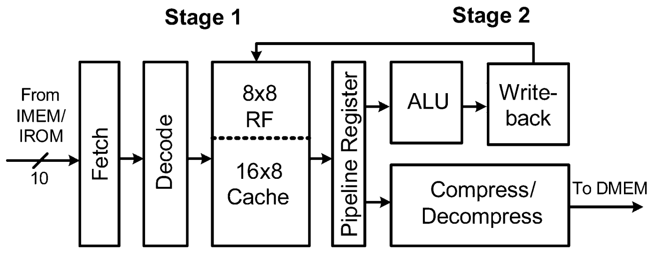


Fig. 4. CPU diagram.

TABLE I
INSTRUCTION SET ARCHITECTURE OVERVIEW

Class	Members	Addressing Mode
Arithmetic	ADD, ADDI, SUB, MOVE, SHR	explicit
Flow Control	BEQZ, JUMPI JUMPR	explicit
Compression	COMP, DECOMP	implicit
	FREE	explicit
Load/Store	LOAD, STORE, STORE_OVER	implicit
Wake	GET_REQ, SEND_REQ, SEND_ACK	implicit
Sleep	HALT	--

efficiency and flexibility in operand specification, two instruction types are available in the Phoenix ISA: *explicit operand* and *implicit operand*. *Explicit operand* instructions use a 3-bit opcode and a 7-bit operand specifier similar to a conventional register-register ISA. As shown in Table I, this instruction format is reserved for arithmetic and flow control instructions, which are used frequently and require flexibility. *Implicit operand* instructions use a 7-bit opcode with a 3-bit modifier and implicitly use special registers R0, R1, and R2 as operand and destination registers. Special instructions like memory load/store and compression/decompression are used infrequently and can therefore use the *implicit operand* format for a small cost.

VI. DMEM Compression for Standby Mode

While efficient instruction encoding helps minimize the footprint of IMEM, we use data compression to help minimize the footprint of DMEM. Along with fine-grained power gating in DMEM (to be discussed in Section VII), compression permits fewer DMEM entries to be retained and enables significant power reductions in standby mode. Compression of instruction and data memories has been explored previously [10], [11]. The IBM Memory Expansion Technology, for example, uses compression to more than double the size of main memory [10] but requires a complex memory management protocol targeted at server systems. To ensure that the energy overheads of compression do not surpass the standby mode reductions of a compressed DMEM, we adopt a simple compression architecture in the Phoenix Processor.

During compression, words from the 16-entry cache are sequentially converted to compressed words using a compression lookup table. The 512-byte virtual memory is divided into 16-byte blocks, and an entire 16-byte block from the cache must be compressed before being sent to the 266-byte physical memory.

The primary function of the Phoenix Processor is sensor data logging, which has two important consequences for compression. The first consequence is that access to memory is largely sequential (since temporally adjacent measurements are stored in spatially adjacent memory locations), thus limiting the compression/decompression overheads associated with random hopping among 16-byte blocks. The other important consequence concerns compression dictionary selection. Typical sensor data is predictably compressible since two temporally adjacent points are likely to differ by only a small amount. The measurement for a particular time can be stored as the difference between the current and previous measurements. The resulting data distribution is tightly distributed around zero, making dictionary selection simpler. Since the Phoenix Processor includes an on-board temperature sensor, we consider a collection of ambient temperature measurements in Michigan as an example [18]. Fig. 5 shows the differences between temporally adjacent temperature measurements for a full year at different sampling intervals. In this difference format, 96% of the data falls in the range -1°C to 1°C assuming a 10 minute sampling interval. We take advantage of this small range by using a fixed compression dictionary that uses short words to represent values in this range and longer words to represent the rare value outside of this range.

We use Huffman encoding to generate a lookup table-based dictionary using temperature measurements from [18] assuming a temperature precision of 1°C and a sampling interval of 30 minutes (which was empirically determined to efficiently compress data sampled at intervals ranging from 5 to 60 minutes). The lookup table converts 8-bit uncompressed data words to

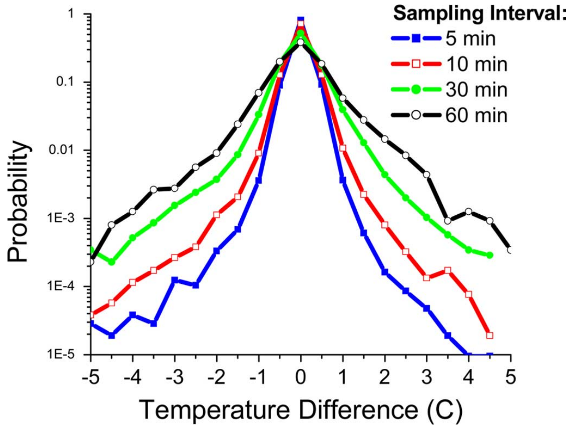


Fig. 5. Distribution of temperature in Muskegon, MI, in 2006 [18] represented as the difference between temporally adjacent measurements.

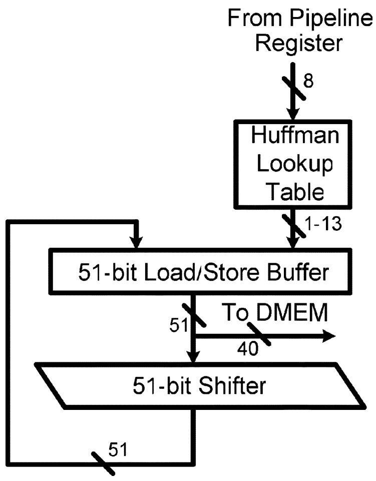


Fig. 6. Hardware support for compression.

compressed words with lengths between 1 and 13 bits. By using a fixed dictionary, the compression operation is simplified significantly, minimizing the active energy penalty. While the footprint of compressed data can grow by up to 60% if measured data is not sufficiently similar to the distribution in [18], an editable fixed dictionary could potentially be stored in DMEM to better match the needs of a specific application.

After using the Huffman lookup table to compress an 8-bit data word, the compressed word is shifted by a 51-bit shifter and then stored in a 51-bit load/store buffer (Fig. 6). Once all entries in a 16-byte block have been loaded in the load/store buffer or the buffer is full, the compressed data is sent to DMEM using one of the three load/store instructions supported by the ISA.

Memory allocation is the primary challenge in implementing compression. Fixed length uncompressed blocks from virtual memory are translated to variable length compressed blocks in physical memory, and efficient placement of the variable length blocks within physical memory can be difficult. To address this problem, we divide DMEM into the two partitions shown in Fig. 7: a statically allocated partition and a dynamically allocated partition. Each 16-byte block in virtual memory is assigned a 40-bit entry in statically allocated memory. Data is normally stored in the statically allocated partition. However, if a

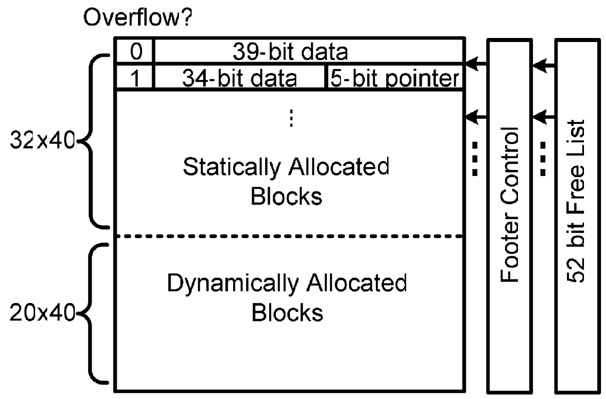


Fig. 7. Memory support for compression.

16-byte block does not fit within its statically allocated entry after compression, the overflow data is stored to an entry in dynamically allocated memory and a 5-bit pointer to the overflow data is stored in the statically allocated entry. A free-list is required to monitor which entries in dynamically allocated memory are available for storage. A priority encoder in the free-list returns the address of the first available entry in the event of an overflow. For compression purposes, the free-list need only monitor the dynamically allocated partition, but we monitor both memory partitions to permit fine-grained power gating (to be discussed in Section VII). Including the overhead of the free-list, the Phoenix Processor compression scheme represents 16-byte blocks with a minimum of 41 bits (a compression ratio of 32%). The effectiveness of the proposed compression scheme will be quantified using test chip measurements in Section IX.

VII. Ultra-Low Standby Power Memory Design

The power consumed by IMEM and DMEM dominates standby mode power since data must be retained in standby mode. In contrast, the CPU and other non-retentive logic can be fully power gated. Minimizing standby power in the IMEM and DMEM is therefore a critical design requirement for the Phoenix Processor. The memories must also be designed for robust operation at low supply voltage to avoid the overhead of a dual supply voltage system.

In the Phoenix Processor, the instruction memory is accessed every cycle by the CPU but does not need to be modified at runtime. Consequently, instruction memory is composed of a 64×10b SRAM (IMEM) and a 64×10b ROM (IROM). Commonly used procedures are stored in IROM while application-specific instructions are stored in IMEM. It is advantageous to put as many instructions in IROM as possible since ROM can be power gated during standby mode. In this work, we use the robust full static CMOS ROM implementation described in [12].

To minimize the standby leakage in retentive cells in IMEM and DMEM, we use the custom ultra-low standby power SRAM cell shown in Fig. 8. The bitcell transistors (cross-coupled inverters and access transistors) use the high-V_{th} IO devices offered by the selected 0.18 μm technology. Although the minimum dimensions of the IO device are larger than those of the thin oxide device, the large leakage reduction justifies the use of the device. We further reduce leakage in the bitcell using stack

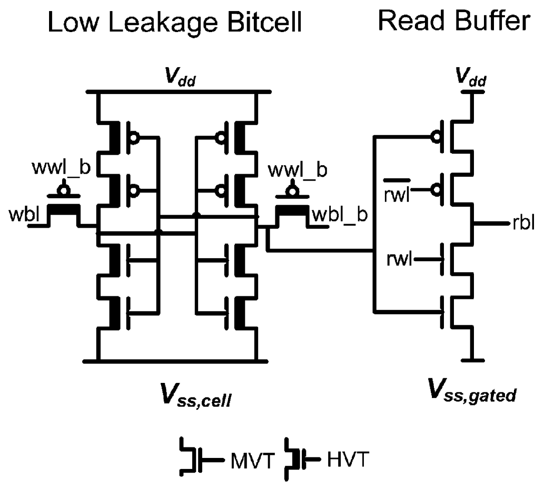


Fig. 8. Proposed ultra-low standby power SRAM cell.

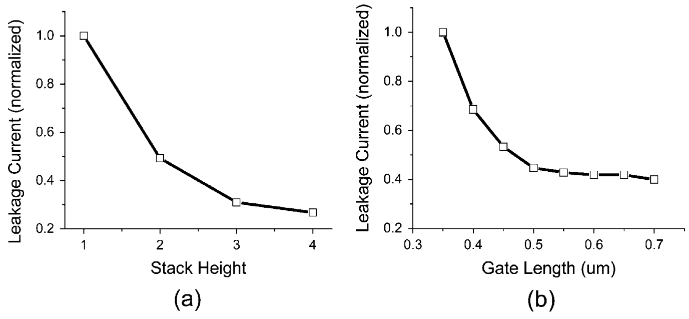


Fig. 9. Effectiveness of (a) stack forcing and (b) gate length biasing for leakage reduction.

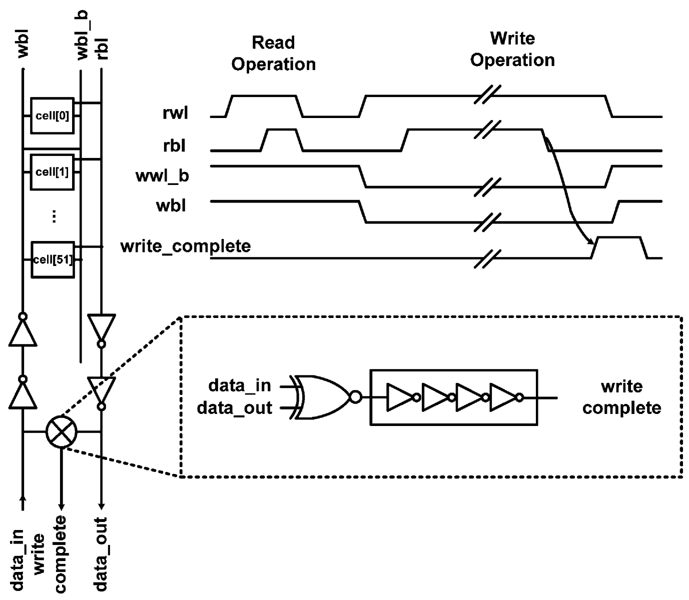


Fig. 10. Memory column diagram showing completion detection.

forcing in the cross-coupled inverters as in other retentive gates. We use a stack height of two because the sensitivity of leakage to stack height becomes linear for larger stacks, as shown in Fig. 9(a). Instead of further stack forcing, we find that increasing the length of the devices in the cross-coupled inverters gives a more area-efficient reduction in leakage. By increasing the length of the transistors from 0.35 μm to 0.50 μm, the leakage is reduced by $\sim 2\times$, as shown in Fig. 9(b). Stack forcing and gate length biasing are not applied to the access transistors to avoid upsetting write margins.

The proposed bitcell enables dramatic leakage reduction at the expense of active power and area. In the target 0.18 μm technology, the area of the proposed cell is 40 μm^2, which is $9.1\times$ larger than the traditional 6 T cell in [24]. Since the instruction and data memory bitcells represent only a fraction of total chip area, the effective area penalty is only $\sim 17\%$. Even with the larger bitcell, measurements show that the memories contribute only 8% of total active power while delivering a $62\times$ reduction in memory standby power (which is 90% of total standby power) as compared to the 6T bitcell in [24]. Given the dominance of standby energy in a typical sensing application (see example in Section I), the proposed bitcell delivers a favorable trade-off.

In addition to operating at low standby power, the proposed SRAM cell in Fig. 8 includes a full swing 4-transistor read buffer for robust low-voltage operation. Read buffers have been previously proposed to decouple read and write margins in low-voltage SRAM cells [13], [14]. The full swing read buffer drives the bitline to both supply rails, ensuring a robust read. Since the read buffer can be power gated in standby mode without upsetting the bitcell data, it is implemented using medium-V_{th} devices for a negligible leakage penalty. The use of medium-V_{th} devices ensures a fast read time comparable to paths in the CPU (implemented with medium-V_{th} devices). This is particularly important for the IMEM, which is accessed every cycle and lies on critical timing paths.

While the read delay is comparable to the delay of the CPU, the write operation through the high-V_{th} devices is slow. We therefore adopt an asynchronous write strategy in which the CPU stalls for 2–3 cycles during the write operation. The DMEM asserts a completion signal to alert the CPU when the write operation is finished. As shown in Fig. 10, the write completion signal is generated by reading the contents of the row being written and comparing to the write data. Since read is single-ended, a replica delays the write completion signal to guarantee that both sides of the cell have been written correctly.

To permit further leakage reduction within DMEM, power gating switches are used to eliminate the standby mode power consumption in non-retentive entries. A particular DMEM entry is power gated only if the free-list (described in Section VI) indicates that the entry is unused. Power gating granularity plays an important role in determining total power. Using a single power gating switch for each row allows the memory to grow to precisely match the footprint of the data. However, the width of a power gating switch is determined by the maximum current needed to read/write a single row, so the width of a single power switch changes minimally with power gating granularity. With one power switch allotted per DMEM row, the total leakage power is sub-optimal because the total power gate width is large. For example, the use of 52 switches for the 52 rows of DMEM would require a total power switch width approximately 52 times wider than the case where one footer is used for the entire DMEM. Higher footer granularity also leads

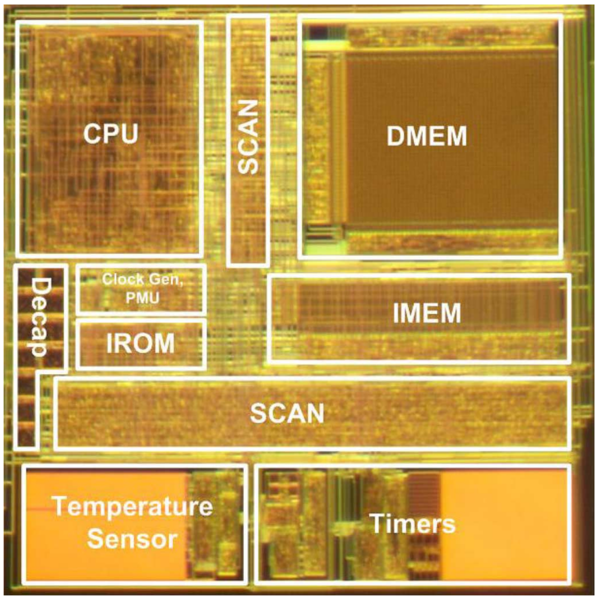


Fig. 11. Phoenix Processor die photo.

to higher complexity in free-list management and a commensurate increase in standby power. We find that minimum standby power is achieved in the Phoenix Processor when two DMEM rows are grouped with a single footer.

VIII. Test Chip Overview

To demonstrate our standby mode strategy, we fabricated the Phoenix Processor in a 0.18 μm process (die photo shown in Fig. 11). The processor includes 60 332 medium-V_{th} devices and 32 167 high-V_{th} devices in an area of 915×915 μm ^{2}. The memory, temperature sensor, and timer blocks were designed using a standard full-custom flow. The CPU and interfaces to memory, temperature sensor, and timer blocks were implemented using both synthesized and semi-custom blocks using a standard tool flow and a library limited to minimum-sized gates with maximum fan-in of three. As in previous low-voltage processors, we routed signal, clock, and power wires using minimum width interconnect to reduce switching energy and improve routing density [5].

IX. Measured Results

A. *Power and Performance Results*

The measured frequency and energy consumed per clock cycle are shown in Fig. 12 as functions of V_{dd} for one test application. The frequency is determined by sweeping the clock frequency, running the test application, and noting the frequency above which the contents of memory are corrupted. The test application runs a short iterative sequence that writes a known list of numbers to DMEM, in the process exercising all timing critical instructions. Power is measured during execution using a high precision ammeter. At the target voltage of 0.5 V, the die highlighted in Fig. 12 operates at 106 kHz with only 2.8 pJ consumed per cycle, which corresponds to only 297 nW.

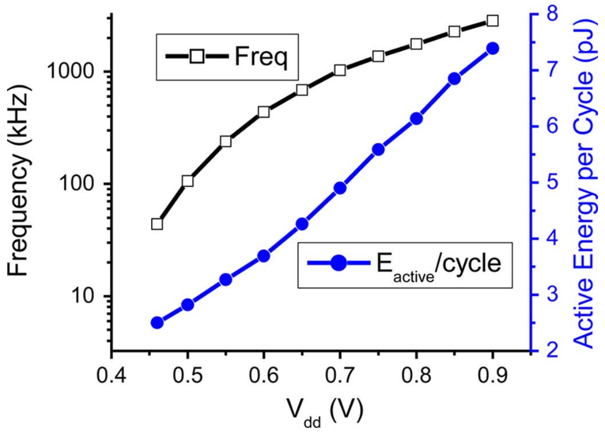


Fig. 12. Measured frequency and energy consumption.

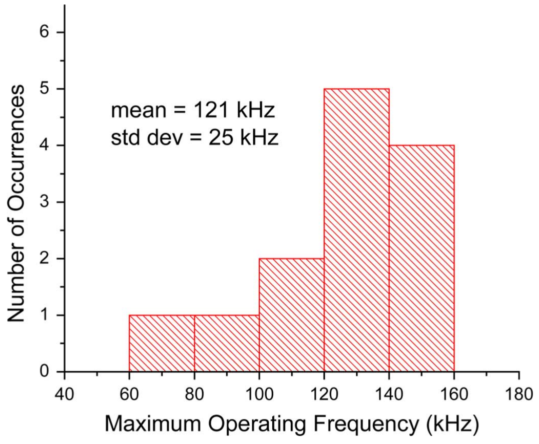


Fig. 13. Measured frequency distribution for 13 dies at $V_{dd} = 0.5$ V.

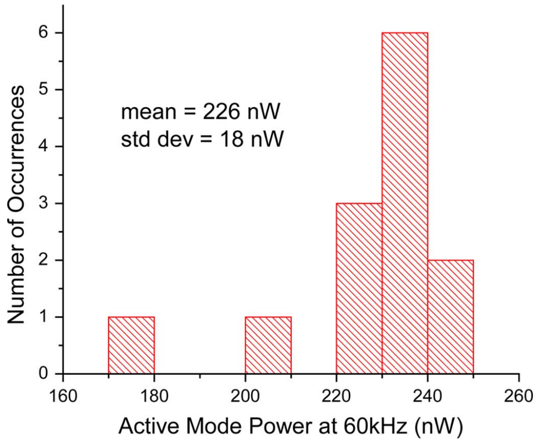


Fig. 14. Measured active mode power distribution at 60 kHz for 13 dies at $V_{dd} = 0.5$ V.

Figs. 13 and 14 show distributions of maximum operating frequency (mean of 121 kHz) and active power at 60 kHz (mean of 226 nW) for 13 dies at $V_{dd} = 0.5$ V. The consequences of variability are particularly important at low operating voltages and have been covered extensively in previous work [5], [21].

Fig. 15 shows that the mean standby mode power consumption for the same 13 dies at $V_{dd} = 0.5$ V is 35.4 pW with 50% of

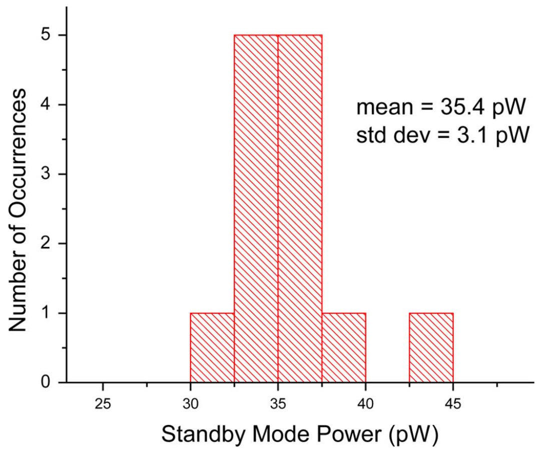


Fig. 15. Measured standby mode power distribution for 13 dies at $V_{dd} = 0.5$ V.

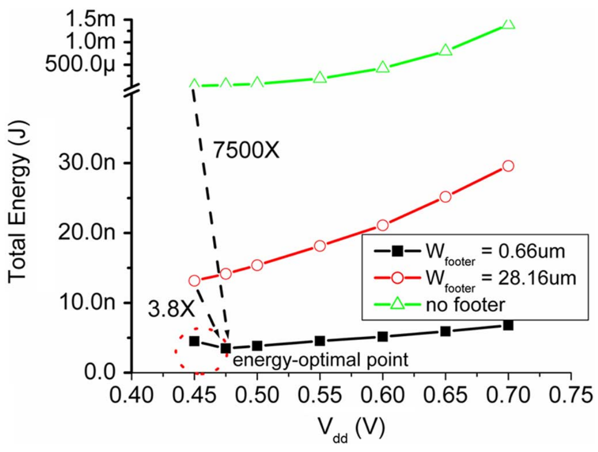


Fig. 17. Total energy consumption assuming 1000 instructions are executed every 10 minutes.

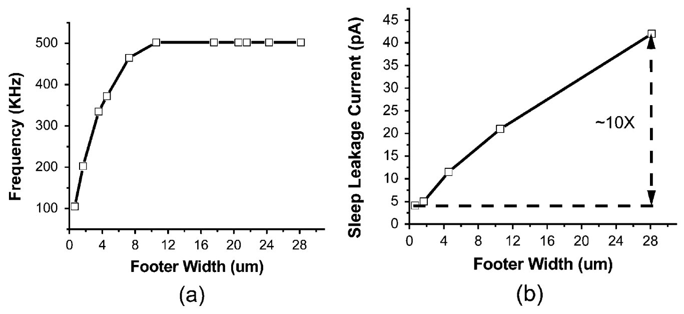


Fig. 16. Measured (a) frequency and (b) standby leakage as functions of CPU footer width.

DMEM entries retained. The IMEM and DMEM consume 89% of standby power while the power gated CPU consumes only 7% of the power. For a typical sensing application in which the sensor remains active for 1000 cycles every 10 minutes, these measurements indicate that the average power consumption is only 42 pW, well within the limit of 177 pW demanded by the battery described in Section I.

B. Power Gating Results

To further investigate our proposed power gating approach, we sweep the footer width on the CPU and measure the energy and performance implications. Fig. 15(a) shows the maximum operating frequency of the CPU as a function of footer width. Frequency reduces by 5× as the footer size approaches the minimum of W $= 0.66$ μm and L $= 0.5$ μm. This performance penalty leads to greater active energy consumption per operation since leakage energy increases with clock period in active mode. The power consumption through the power gating switch results in an additional energy penalty. However, the standby leakage power savings from the narrow footer width, shown in Fig. 15(b), easily offsets these penalties and reduces the total energy for the Phoenix Processor. Fig. 16 confirms that the total energy consumption is 3.8× lower for the small footer (W $= 0.66$ μm) than the large footer (W $= 28$ μm) assuming 1000 instructions are executed every 10 minutes. The small footer saves several orders of magnitudes of total energy compared to a design with no power switch.

C. Memory Results

The IMEM and DMEM consume 7.1 fW/bitcell (not including the overhead of decoders and row drivers). The IMEM alone accounts for 39% of the total standby power (including the overhead of decoders and drivers), which underscores the importance of instruction set optimization. If the instruction width had been set to 15 bits instead of 10 bits, measurements of a typical die show that the standby power of IMEM would have increased by 20%, which equates to ~ 8% increase in total standby power.

Unlike IMEM, the power consumed by DMEM, which amounts to 51% of total standby power at 50% retention, can be reduced significantly by compressing the data and by changing the number of DMEM entries retained to match the footprint of compressed memory. For a typical die, the DMEM consumes 22 pW with all entries retained and 7.5 pW with all entries power gated due to the overhead of maintaining the free-list (i.e., the overhead of data compression).

To quantify the system-level benefits of compression and fine-grained power gating in memories, consider an ambient temperature sensing application in which the Phoenix Processor wakes up once every 10 minutes and runs a 1000 cycle routine to measure temperature and store the measured data. Using the measured temperature dependence of frequency and power consumption shown in Fig. 18 and a subset of the temperature profile from [18], we can compute the reduction in energy due to compression and power gating in DMEM over the lifetime of the chip. For this case study we assume that temperature is measured to a precision of 1°C. Fig. 19(a) shows the total energy consumed over 37 hours (the time period over which uncompressed memory fills to capacity) for 3 cases. In Case 1, neither compression nor power gating are used in DMEM. In Case 2, power gating is used, but compression is not used. Finally, in Case 3, both power gating and compression are used. The use of power gating within DMEM (Case 2) reduces energy consumption by 7.3%, and the use of both power gating and compression (Case 3) reduces energy consumption by 14.7%. Compression also increases the effective size of DMEM and enables the processor to remain active for 7.8× longer before memory fills to capacity, as shown in Fig. 19(b).

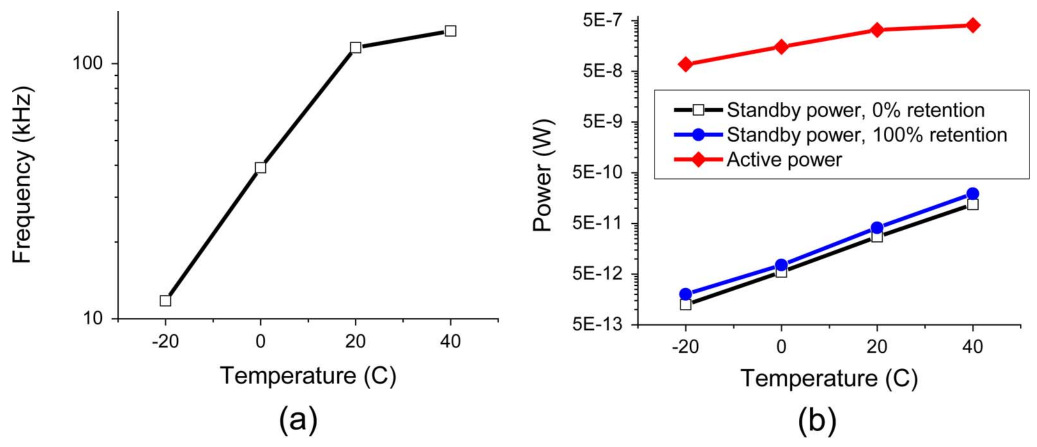


Fig. 18. Measured (a) frequency and (b) power as functions of temperature.

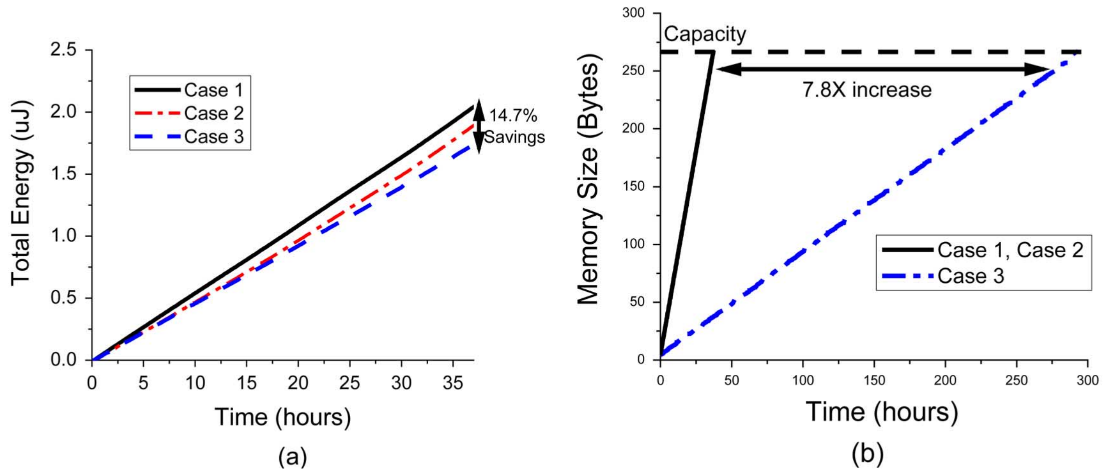


Fig. 19. Computed time profiles of (a) energy and (b) memory size for a temperature measurement routine.

X. Conclusion

Cubic millimeter computing will bring new computational capacities to a wide range of applications. In this work, we describe a sensor processor that operates at $V_{\mathrm{dd}} = 0.5$ V to minimize active mode energy and uses device-, circuit-, and architecture-level techniques to minimize standby mode energy. Measurements show that Phoenix consumes 226 nW in active mode and only 35.4 pW in standby mode. A thin film battery with the same form factor as Phoenix could provide a lifetime on the order of years, making Phoenix an attractive candidate for future cubic millimeter computing systems.

References

[1] B. Warneke, M. Last, B. Liebowitz, and K. Pister, "Smart dust: Communicating with a cubic-millimeter computer," *Computer*, vol. 34, no. 1, pp. 44–51, 2001.

[2] G. Ono, T. Nakagawa, R. Fujiwara, T. Norimatsu, T. Terada, M. Miyazaki, K. Suzuki, K. Yano, Y. Ogata, A. Maeki, S. Kobayashi, N. Koshizuka, and K. Sakamura, "1-cc computer: Cross-layer integration with 3.4-nW/bps link and 22-cm locationing," in *Symp. VLSI Circuits Dig.*, 2007, pp. 90–91.

[3] U. Schnakenberg, P. Walter, G. vom Bogel, C. Kruger, H. C. Ludtke-Handjery, H. A. Richter, W. Specht, P. Ruokonen, and W. Mokwa, "Initial investigations on systems for measuring intraocular pressure," *Sensors and Actuators*, vol. 85, no. 1–3, pp. 287–291, 2000.

[4] Y.-S. Lin, S. Hanson, F. Albano, C. Tokunaga, R. Haque, K. Wise, A. Sastry, D. Blaauw, and D. Sylvester, "Low-voltage circuit design for widespread sensing applications," in *Proc. Int. Symp. Circuits and Systems (ISCAS)*, 2008, pp. 2558–2561.

[5] S. Hanson, B. Zhai, M. Seok, B. Cline, K. Zhou, M. Singhal, M. Minuth, J. Olson, L. Nazhandali, T. Austin, D. Sylvester, and D. Blaauw, "Exploring variability and performance in a sub-200 mV processor," *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 881–891, Apr. 2008.

[6] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," in *Proc. Design Automation Conf.*, 2004, pp. 868–873.

[7] B. Zhai, L. Nazhandali, J. Olson, A. Reeves, M. Minuth, R. Helfand, S. Pant, D. Blaauw, and T. Austin, "A 2.60 pJ/Inst subthreshold sensor processor for optimal energy efficiency," in *Symp. VLSI Circuits Dig.*, 2006, pp. 154–155.

[8] A. Wang and A. Chandrakasan, "A 180 mV FFT processor using subthreshold circuit techniques," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, 2004, pp. 292–293.

[9] M. Seok, S. Hanson, Y.-S. Lin, Z. Foo, D. Kim, Y. Lee, N. Liu, D. Sylvester, and D. Blaauw, "The Phoenix Processor: A 30 pW platform for sensor applications," in *Symp. VLSI Circuits Dig.*, 2008, pp. 188–189.

[10] R. B. Tremaine, P. A. Franaszek, J. T. Robinson, C. O. Schulz, T. B. Smith, M. E. Wazlowski, and P. M. Bland, "IBM memory expansion technology (MXT)," *IBM J. Research and Development*, vol. 45, no. 2, pp. 271–286, 2001.

[11] C. Lefurgy, P. Bird, I. Chen, and T. Mudge, "Improving code density using compression techniques," in *Proc. Int. Symp. Microarchitecture*, 1997, pp. 194–203.

[12] M. Seok, S. Hanson, J. Seo, D. Sylvester, and D. Blaauw, "Robust ultra-low voltage ROM design," in *Proc. IEEE Custom Integrated Circuits Conf. (CICC)*, 2008, pp. 423–426.

[13] L. Chang, D. Fried, J. Hergenrother, J. Sleight, R. Dennard, R. Montoye, L. Sekaric, S. McNab, A. Topol, C. Adams, K. Guarini, and W. Haensch, "Stable SRAM cell design for the 32 nm node and beyond," in *Symp. VLSI Technology Dig.*, 2005, pp. 128–129.

[14] B. Calhoun and A. Chandrakasan, "A 256 kb sub-threshold SRAM in 65 nm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, 2006, pp. 2592–2593.

[15] B. Zhai, D. Blaauw, D. Sylvester, and S. Hanson, "A sub-200 mV 6T SRAM in 0.13 μm CMOS," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, 2007, pp. 332–333.

[16] M. Seok, S. Hanson, D. Sylvester, and D. Blaauw, "Analysis and optimization of sleep modes in subthreshold circuit design," in *Proc. Design Automation Conf.*, 2007, pp. 694–699.

[17] S. Narendra, S. Borkar, V. De, D. Antoniadis, and A. Chandrakasan, "Scaling of stack effect and its application for leakage reduction," in *Proc. Int. Symp. Low Power Electronics and Design (ISLPED)*, 2001, pp. 195–200.

[18] "Muskegon meteorological data," Great Lakes Environmental Research Laboratory, National Oceanic and Atmospheric Administration Aug. 20, 2008 [Online]. Available: http://www.glerl.noaa.gov/metdata/mkg/archive/

[19] M. Seok, D. Sylvester, and D. Blaauw, "Optimal technology selection for minimizing energy and variability in low voltage applications," in *Proc. Int. Symp. Low Power Electronics and Design (ISLPED)*, 2008, pp. 9–14.

[20] S. Hanson, M. Seok, D. Sylvester, and D. Blaauw, "Nanometer device scaling in subthreshold circuits," in *Proc. Design Automation Conf.*, 2007, pp. 700–705.

[21] J. Kwong, Y. Ramadass, N. Verma, M. Koesler, K. Huber, H. Moormann, and A. Chandrakasan, "A 65 nm sub-V_t microcontroller with integrated SRAM and switched-capacitor DC-DC converter," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, 2008, pp. 318–319.

[22] L. Nazhandali, B. Zhai, J. Olson, A. Reeves, M. Minuth, R. Helfand, S. Pant, T. Austin, and D. Blaauw, "Energy optimization of subthreshold-voltage sensor network processors," in *Proc. Int. Symp. Computer Architecture*, 2005, pp. 197–207.

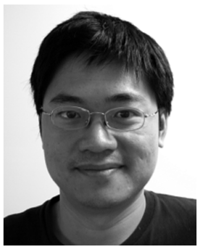
[23] M. Sheets, F. Burghardt, T. Karalar, J. Ammer, Y. H. Chee, and J. Rabaey, "A power-managed protocol processor for wireless sensor networks," in *Symp. VLSI Circuits Dig.*, 2006, pp. 212–213.

[24] C. H. Diaz *et al.*, "A 0.18 μm CMOS logic technology with dual gate oxide and low-k interconnect for high-performance and low-power applications," in *Symp. VLSI Technology Dig.*, 1999, pp. 11–12.

[25] H. Kawaguchi, K. Nose, and T. Sakurai, "A CMOS scheme for 0.5 V supply voltage with pico-ampere standby current," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, 1998, pp. 192–193.

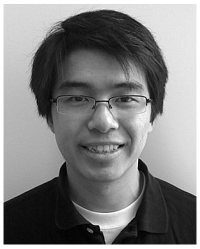
Scott Hanson (S'05) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Michigan, Ann Arbor, in 2004, 2006, and 2009, respectively. As a Ph.D. student he was the recipient of an SRC fellowship. He is currently a research fellow in electrical engineering at the University of Michigan. His research interests include low voltage circuit design for ultra-low energy applications, variation tolerant circuit design, and energy-efficient high performance circuit design.

Mingoo Seok (S'06) was born in Seoul, Republic of Korea. He received the Bachelors degree in electrical engineering from Seoul National University, Republic of Korea, in 2005. He is currently working towards the Ph.D. degree in electrical engineering at University of Michigan, Ann Arbor. His research interests include low-voltage/low-power circuit design and physical aspects of submicron technology. He is the recipient of a KFAS fellowship.



Yu-Shiang Lin (S'04–M'08) received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2000 and 2002, respectively. In 2008, he received the Ph.D. degree in electrical engineering from the University of Michigan, Ann Arbor.

Since 2008, he has been with the IBM T. J. Watson Research Center, Yorktown Heights, NY, where he is a Postdoctoral Researcher. His research has focused on ultra-low-power VLSI circuits design.



ZhiYoong Foo received the B.S. degree in electrical engineering from the University of Michigan, Ann Arbor, in 2006, where he is currently working towards the M.S. degree in electrical engineering. His current research interests include VLSI design with particular emphasis on low power and high performance.

Daeyeon Kim (S'08) received the B.S. degree in electronic and electrical engineering from Pohang University of Science and Technology (POSTECH), Pohang, Korea, in 2006, and the M.S. degree in electrical engineering from the University of Michigan, Ann Arbor, in 2008, where he is currently working towards the Ph.D. degree. His research interests include low-power/variation-tolerant SRAM design and ultra-low-power applications beyond CMOS. He is the recipient of a KFAS Fellowship.

Yoonmyung Lee (S'08) received the B.S. degree in electrical engineering (*summa cum laude*) from the Pohang University of Science and Technology (POSTECH), Pohang, Korea, in 2004, and the M.S. degree in electrical engineering from the University of Michigan, Ann Arbor, in 2008, where he is currently working towards the Ph.D. degree.

In summer 2008, he held a research intern position with the Circuit Research Lab, Intel, Hillsboro, OR. His current research interests include low power, high performance VLSI circuit design and low leakage memory design.

Mr. Lee was a recipient of a Presidential award for outstanding scholastic achievement during his undergraduate coursework.

Nurrachman Liu (S'05) was born in Taipei, Taiwan, and grew up in Canada. He received the B.S. degree in electrical engineering and computer sciences from the University of California, Berkeley, in 2006, and the M.S. degree in electrical engineering and computer sciences from the University of Michigan, Ann Arbor, in 2008, where he is currently working towards the Ph.D. degree. His research interests include low-voltage circuit design, variation-tolerant circuit design, and energy-efficient mixed-signal circuit design.

Dennis Sylvester (S'95–M'00–SM'04) received the B.S. degree in electrical engineering (*summa cum laude*) from the University of Michigan, Ann Arbor, and the Ph.D. degree in electrical engineering from the University of California, Berkeley, in 1999. His dissertation research was recognized with the David J. Sakrison Memorial Prize as the most outstanding research in the UC-Berkeley EECS Department.

He is now an Associate Professor of electrical engineering and computer science at the University of Michigan, Ann Arbor. He previously held research staff positions in the Advanced Technology Group of Synopsys, Mountain View, CA, Hewlett-Packard Laboratories in Palo Alto, CA, and a visiting professorship in electrical and computer engineering at the National University of Singapore. He has published numerous articles along with one book and several book chapters in his field of research, which includes low-power circuit design and design automation techniques, design-for-manufacturability, and interconnect modeling. He also serves as a consultant and technical advisory board member for several electronic design automation and semiconductor firms in these areas.

Dr. Sylvester received an NSF CAREER award, the Beatrice Winner Award at ISSCC, an IBM Faculty Award, an SRC Inventor Recognition Award, and several best paper awards and nominations. He is the recipient of the ACM SIGDA Outstanding New Faculty Award and the University of Michigan Henry Russel Award for distinguished scholarship. He has served on the technical program committee of numerous design automation and circuit design conferences, the steering committee of the ACM/IEEE International Symposium on Physical Design, and was general chair for the 2005 ACM/IEEE Workshop on Timing Issues in the Synthesis and Specification of Digital Systems (TAU). He is currently an Associate Editor for IEEE TRANSACTIONS ON CAD and previously served as Associate Editor for IEEE TRANSACTIONS ON VLSI SYSTEMS. He is a member of ACM, American Society of Engineering Education, and Eta Kappa Nu.

David Blaauw (M'94–SM'07) received the B.S. degree in physics and computer science from Duke University, Durham, NC, in 1986, and the Ph.D. degree in computer science from the University of Illinois, Urbana, in 1991.

Until August 2001, he worked for Motorola, Inc. in Austin, TX, where he was the Manager of the High Performance Design Technology group. Since August 2001, he has been on the faculty at the University of Michigan, Ann Arbor, as an Associate Professor. His work has focused on VLSI design and CAD with particular emphasis on circuit design and optimization for high-performance and low-power applications.

Dr. Blaauw was the Technical Program Chair and General Chair for the International Symposium on Low Power Electronic and Design and was the Technical Program Co-Chair and member of the Executive Committee the ACM/IEEE Design Automation Conference. He is currently a member of the ISSCC Technical Program Committee.