

# Energy Optimization of Subthreshold-Voltage Sensor Network Processors

Leyla Nazhandali, Bo Zhai, Javin Olson, Anna Reeves, Michael Minuth,  
Ryan Helfand, Sanjay Pant, Todd Austin and David Blaauw  
*The University of Michigan*  
1301 Beal Ave, Ann Arbor, MI 48109  
subliminal@eecs.umich.edu

## Abstract

*Sensor network processors and their applications are a growing area of focus in computer system research and design. Inherent to this design space is a reduced processing performance requirement and extremely high energy constraints, such that sensor network processors must execute low-performance tasks for long durations on small energy supplies. In this paper, we demonstrate that subthreshold-voltage circuit design (400 mV and below) lends itself well to the performance and energy demands of sensor network processors. Moreover, we show that the landscape for microarchitectural energy optimization dramatically changes in the subthreshold domain. The dominance of leakage power in the subthreshold regime demands architectures that i) reduce overall area, ii) increase the utility of transistors, while iii) maintaining acceptable CPI efficiency. We confirm these observations by performing SPICE-level analysis of 21 sensor network processors and memory architectures. Our best sensor platform, implemented in 130nm CMOS and operating at 235 mV, only consumes 1.38 pJ/instruction, nearly an order of magnitude less energy than previously published sensor network processor results. This design, accompanied by bulk-silicon solar cells for energy scavenging, has been manufactured by IBM and is currently being tested.*

## 1. Introduction

Sensor network processing is emerging as a new frontier of computer system design. Sensor network processors combine sensing, computation, and communication into small battery-powered form factors that can be placed into the environments that they monitor [8, 15]. Sensor networking applications span a vast range, from medical monitoring applications, to environmental sensing, to industrial inspection, and military surveillance [7].

Sensor platforms carry with them a number of form factor requirements that place heavy constraints on the energy available for computation [12, 15]. First, many applications

require a sensor node that is very small in size. For example, an eyeball activity monitor must be small enough to be embedded into the epidermis of the eyeball. Second, sensor network processors must carry their energy supplies within this small form factor, in the form of batteries or apparatus appropriate to scavenge energy, such as a solar cell. In either case, the quantity of energy available to sensor application processing is quite limited. For example, a 2g vanadium oxide battery contains 720 mA-hr of energy, enough to power ARM, Ltd's most energy-efficient ARM 720T processor at 100MHz for 45 hrs [1]. Certainly, this energy payload is not sufficient for long-term sensing applications, such as a heart monitor for which installation requires surgery.

Fortunately, the energy demands of sensor processing platforms are mitigated by their modest processing demands [6, 10]. For example, a blood pressure monitor sensor requires a sensing capability of approximately 800 bps. Passing the sensing data to a software-based digital threshold monitor, which watches for high or low blood pressure events, would require about 10,000 instructions per second processing power. Higher-rate natural data streams, such as electrical signals from the human brain, are generated at data rates of about 3,200 bps. Even these higher-rate signals could be processed by a digital filter, analyzed with a threshold monitor and compressed for storage with less than 56,000 instructions per second. Given the low computational demands of many sensor networking applications, there is a significant opportunity to reduce processing energy demands through low-frequency, low-voltage designs.

### 1.1. Contributions of This Paper

In this paper, we explore the landscape of architectural energy optimization for low- to mid-bandwidth sensor network processing demands. We find that superthreshold ( $V_{dd} > V_{th}$ ) circuit implementations are too fast and energy-hungry for even the most simple of microarchitectures, leading us to the exploration of subthresh-

old ( $V_{dd} < V_{th}$ ) circuit implementations. Additionally, we find the landscape of energy optimization for subthreshold designs to be much more treacherous than that of superthreshold design. Specifically, we find that:

- *Area must be minimized* as it is a critical energy factor due to the dominance of leakage energy at subthreshold voltages.
- *Transistor utility must be maximized* because effective transistor computation offsets static leakage power, which permits a lower operating voltage and lower overall energy consumption for the design.
- *CPI must be minimized* at the same time, otherwise, gains through small area and high transistor utility are squandered on inefficient computation.

To address area concerns we design a nibble-sized (4 bits) variable-length instruction set, with a variety of optimizations to reduce code size. We find that our best ISA design significantly improves code density, while only slightly aggravating the size of the processor control logic.

Additionally, we examine 21 different microarchitectural designs with varied datapath widths, degree of pipelining, prefetching capability, and with varied register and memory architectures. To achieve the fidelity necessary to evaluate their energy efficiency and performance at subthreshold voltages, we produce a layout of each design in IBM 0.13 $\mu$ m fabrication technology. We confirm our observed tenets of subthreshold design: simple but CPI-efficient designs with high transistor utility and conservative area yield the most energy efficiency. Our most energy-efficient sensor platform is a simple processor design with an 8-bit datapath, running at 182kHz at 235mV and consuming 1.4pJ/instruction. We also find that many of our area- and performance-optimal designs at subthreshold voltage levels are not the best performing designs at superthreshold voltages, confirming our observation that the trade-offs surrounding energy-efficient design are dramatically different in the subthreshold voltage domain.

We have completed a prototype physical design of our most energy-efficient subthreshold-voltage sensor network processor. We briefly describe the design, and the accompanying infrastructure on the test chip, which includes bulk-silicon solar cells, experimental memory designs, and test harnesses. Our prototype chip has been manufactured by IBM in 0.13 $\mu$ m technology, and it is currently being tested.

The remainder of this paper is organized as follows. Section 2 introduces our sensor networking applications, representative data streams, and then makes a case for why sensor network processors should employ subthreshold-voltage circuit implementations. Section 3 introduces subthreshold circuit design and highlights the complexities of energy optimization at ultra-low voltages. Section 4 presents studies we performed to design our sensor network proces-

sor instruction set. Additionally, microarchitectural trade-off studies were performed to determine which combination of features best minimizes energy while meeting sensor processing performance demands. Finally, Section 5 draws conclusions and gives insights for future sensor network processor designs, along with presenting highlights of our prototype sensor network processor test chip.

## 2. Sensor Network Processing

To effectively gauge the processing and energy demands of sensor network processors, we must first assemble a sensor network processing benchmark collection and examine the microprocessors’ performance under a variety of sensor processing data streams. Table 1 lists the sensor network processing benchmarks we examine in this study. The applications are divided into three categories: communication algorithms, computational processing, and sensing algorithms. These programs represent a broad slice of the types of applications one could expect to see on an ultra-low energy sensor network processor platform.

Applica- tion	Description	Code Size nibble
<i>Communication Algorithms</i>		
adRout	Ad-hoc router control algorithm	42
compRLE	Run-length encoded compressor	73
TEA	TEA encryption algorithm	85
crc8	Cyclic redundancy code generator	99
<i>Computational Processing</i>		
divide	Unsigned integer division	80
multiply	Unsigned multiplication	48
inSort	In-place insertion sort	78
binSearch	Binary search	90
<i>Sensing Algorithms</i>		
intAVG	Signed integer average	113
intFilt	4-tap signed FIR filter	106
tHold	Digital threshold detector	45

**Table 1. Sensor Network Processing Algorithms.**

In the communication domain, *adRout* represents a simple routing routine for an ad-hoc sensor communication network (similar to [13]). The algorithm accepts packets from nearby nodes and determines if the packet should be dropped or be re-sent based on whether or not the destination node is closer or further away from the sender node. The *compRLE* [4] represents a low-overhead compression algorithm, which is typically applied to data packets before transmission. *TEA* [16] is a 128-bit strong encryption algorithm, similar to what would be used in secure sensing ap-

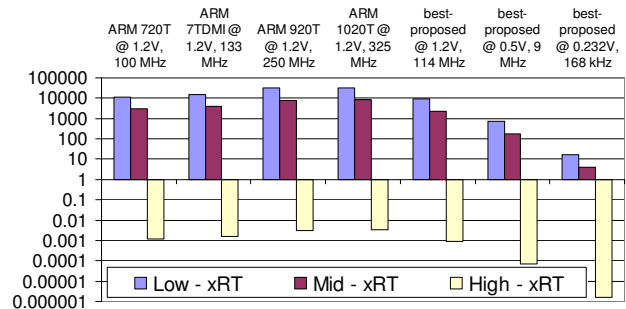
Phenomena	Sample Rate (in Hz)	Sample Prec. (in bits)
<i>Low Frequency Band (&lt; 100Hz)</i>		
Ambient light level	0.017 - 1	16
Atmospheric temperature	0.017 - 1	16
Ambient noise level	0.017 - 1	16
Barometric pressure	0.017 - 1	8
Wind direction	0.017 - 1	8
Body temperature	0.1 - 1	8
Natural seismic vibration	0.2 - 100	8
Heart rate	0.8 - 3.2	1
Wind speed	1 - 10	16
Oral-nasal airflow	16 - 25	8
Blood pressure	50 - 100	8
<i>Mid Frequency Band (100Hz - 1000Hz)</i>		
Engine temperature & pressure	100 - 150	16
EEG (brain electrical activity)	100 - 200	16
EOG (eyeball electrical activity)	100 - 200	16
ECG (heart electrical activity)	100 - 250	8
<i>High Frequency Band (&gt; 1kHz)</i>		
Breathing sounds	100 - 5k	8
EMG (skeletal muscle activity)	100 - 5k	8
Audio (human hearing range)	15 - 44k	16
Video (digital television)	10M	16
Fast A/D conversion	1G	8

**Table 2. Sensor Processing Data Rates. Sensor processing applications are divided into low, medium, and high-bandwidth processing demands.**

plications. Finally, *crc8* calculates an 8-bit checksum for a 24-bit piece of data, appending the checksum to the end of the data to produce a 32-bit value. This particular CRC can detect up to 8 consecutive wrong bits. In the computation processing domain, we have integer multiply and integer divide algorithms. The computational workload also includes insertion sort and binary search algorithms that have many possible uses in sensor applications. For example, sorting is used in sensing applications where the top-N samples are tabulated. In the sensing domain, we have data averaging and filtering algorithms in addition to a Schmidt trigger threshold detector designed to spot events where data values fall below or above a specified threshold (with a hysteresis).

Sensor network platforms evaluate environmental information in real-time, by reading, processing, compressing, storing, and eventually transmitting the information to interested parties. To better understand the computational demands of a real-time sensor network platform, we tabulated the data processing rates of a variety of phenomena. Table

2 lists a number of applications and their associated sample rates (in Hz, samples per second) and the sample precision (in bits per sample). These data rates were gathered from a variety of sources, including [3, 2, 5]. We have divided the applications into *low*-, *mid*- and *high*-bandwidth rates, which reflect sample rates of less than 100 Hz, 100 – 1 kHz, and greater than 1 kHz, respectively.



**Figure 1. Performance of Sensor Network Processor Applications on Embedded Targets. xRT ratings for four commercial processors and an energy-efficient design proposed in this paper at three different voltages with respect to low, mid and high-bandwidth requirements. The performance metric xRT indicates how many times faster than real time a processor performs.**

Figure 1 illustrates the performance of four commercial embedded processors, in addition to one energy-efficient sensor network processor design proposed in this paper at three different voltages. Each of the processors are implemented in a  $0.13\mu\text{m}$  IBM process. For each processor we show the *xRT* rating, which is computed via simulation by determining how many times faster than real-time the processor can handle the worst-case data stream rate on the most computationally intensive sensor benchmark. For example, the ARM 720T at 1.2V with a 100 MHz clock is able to process worst-case mid-bandwidth data 2965 times faster than real-time data rates.

A few of the high-bandwidth sensor applications can be served by the commercial ARM processors, while the highest bandwidth A/D sample rate greatly exceeds the computation capability of even the most competent embedded processors. Consequently, we restrict our studies in this paper to the lesser demands of the low- and mid-bandwidth sensor network applications. It is clear from Figure 1 that the low- and mid-bandwidth sensor processing applications have computational demands that are well below those delivered by the commercial ARM processors. The same is true for the energy-efficient proposed design at full-voltage (1.2V) and 114 MHz. This design services the mid-bandwidth applications at more than 2,253 times the required worst-case processing requirement.

We can reduce the energy demands of these applications by reducing the frequency of the processor, which in turn accommodates reductions in the voltage. As voltage is lowered, energy demands will decrease quadratically. However, even the lowest superthreshold voltages still deliver too much performance. The energy-efficient proposed design is shown in Figure 1 at 0.5V (*i.e.*, the lowest superthreshold voltage accommodated by the IBM process technology) and runs with a 9 MHz clock. Even this low-voltage design is capable of delivering 180 times the performance required by the low- and mid-range sensor processing applications.

To further reduce energy requirements, we must consider running our sensor network processors at subthreshold voltages. At subthreshold voltages the processor will operate with a  $V_{dd}$  below that of  $V_{th}$ , resulting in a significant energy reduction with a great impact on performance. The energy-efficient subthreshold design in Figure 1 delivers more than 4 times the desired performance for mid-bandwidth applications at 232 mV with a 168kHz clock. Running this design any slower would require additional energy – why this is the case, we expound in the following section.

It is noteworthy to mention that even increasing the sleep time of the processors is not helpful in reducing the energy per instruction. The run-and-sleep technique, in which the processor runs to execute a job and goes to sleep when the job is finished, reduces the overall energy consumption of a processor because it saves the energy consumed in idle state. However, in our analysis we are considering energy per instruction; hence, not including the idle energy consumption. In other words, we are making a comparison between the energy consumption of the processors during their service time, and assume they all employ some technique to save energy in idle periods.

### 3. Subthreshold-Voltage Circuit Design

Dynamic voltage scaling has been a very effective method for improving the energy efficiency of processors which are not performance constrained. However, as discussed in the previous section, the applications considered in this paper require performance levels that are still orders of magnitude less than that of a network processor scaled to the lower limit of the traditional dynamic voltage scaling range. This lower limit has typically been restricted to approximately  $V_{dd}/2$ , and is imposed upon by a few sensitive circuits with analog-like operation, such as sense-amplifiers and phase-locked loops. However, it has been well known for some time that standard CMOS gates operate seamlessly from full  $V_{dd}$  to well below the threshold voltage, at times reaching as low as 100mV [9]. With careful design, it is possible to ad-

dress the voltage scaling limit of more sensitive components. For instance, by replacing them with more conventional CMOS based implementations, it is possible to construct processor designs that operate well below the threshold voltage. Recently, a number of such prototype designs have been demonstrated [9, 17, 14].

Subthreshold design raises a number of circuit-level design issues, including increased sensitivity to process variations, soft error strikes, and robust memory and PLL designs. We are currently investigating new methods to address these particular issues and will report results after more testing of our prototype chip. In this paper, we restrict our discussion to the issue of architectural energy-efficient design at subthreshold voltages. We address two issues:

- *Determination of the energy-optimal operating voltage.* At superthreshold operation, reducing the supply voltage always improves the energy efficiency. At subthreshold operation, this is not true, as leakage energy increases with voltage scaling and hence a supply voltage exists where energy per instruction is minimized.
- *Identification of design parameters which determine the energy efficiency of a design when operating at the energy-optimal supply voltage.* The understanding of these parameters is key to designing energy-efficient architectures for subthreshold operation. In addition, we point out how these parameters differ from the critical factors considered at superthreshold operation.

In the next section we discuss the operation of a CMOS gate in subthreshold operation and present the expressions that govern its energy and delay characteristics.

#### 3.1. Subthreshold-Voltage Circuit Operation

The transistors of a CMOS gate, operating at superthreshold supply voltages, effectively function like switches. When the input of the inverter shown in Figure 2 is  $V_{dd}$ , the NMOS transistor is strongly conducting while the PMOS transistor is in cut-off, resulting in 0V at the gate output. The delay of the gate is proportional to the current supplied by the conducting NMOS transistor, which is referred to as the on-current,  $I_{on}$ . Furthermore, the delay of the gate scales approximately linearly for voltages in the superthreshold regime [17].

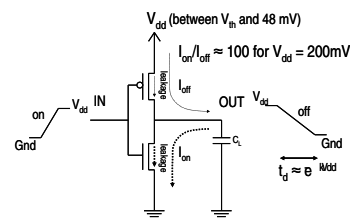


Figure 2. Inverter at Subthreshold Voltage.

However, even with a gate to source voltage ( $V_{gs}$ ) of 0V, the PMOS transistor is not completely turned off but allows a small leakage current to exist, referred to as subthreshold drain-to-source leakage current or the off-current,  $I_{off}$ . This leakage current does not significantly influence the logic functionality or delay of the gate in superthreshold operation because the conducting NMOS transistor is many orders of magnitude stronger, resulting in an  $I_{on}/I_{off}$  current ratio of approximately 10,000 or more.

If the supply voltage is reduced below the threshold voltage, both the NMOS and PMOS transistors are in cut-off, regardless of the logic value of the inverter input. In this case, both transistors exhibit subthreshold current. However, the subthreshold leakage current is an exponential function of  $V_{gs}$ , which forms the basis for the operation of the gate in this operating regime. For instance, if the supply voltage is 200 mV and the input of the inverter is at  $V_{dd}$ , the NMOS transistor will have a  $V_{gs} = 200mV$  while the PMOS transistor has  $V_{gs} = 0V$ . In current technologies, the dependence of leakage current on  $V_{gs}$  is approximately one decade per 100mV of  $V_{gs}$  and hence, the NMOS transistor will have approximately 100 times the leakage current of the PMOS transistor. The difference in the leakage current of the two transistors provides the drive current for discharging the output capacitance that results in the signal transition. Furthermore, the  $I_{on}/I_{off}$  ratio is approximately 100, which is still sufficiently high to obtain an inverter output voltage swing that is nearly rail-to-rail.

However, if we reduce the supply voltage from 200mV to 100mV, the leakage current of the NMOS transistor, when the input of the inverter is  $V_{dd}$ , exponentially reduces, to only 10 times that of the PMOS transistor. Hence, the delay of the inverter is increased by 10x for a 2x reduction in supply voltage. Therefore, the exponential dependence of leakage current on  $V_{gs}$  results in an exponential dependence of circuit delay on supply voltage as shown in the following simple expression:

$$t_{clk} \propto e^{-kV_{dd}}$$

where  $k$  is a technology and temperature dependent constant. In addition, the reduction of the supply voltage to 100mV has reduced the  $I_{on}/I_{off}$  ratio to only 10, resulting in a compressed output voltage swing. As the supply voltage is reduced further, it is clear that the output voltage swing will reduce to the point where it can no longer encode a logic value. It was previously shown that this minimum functional supply voltage is approximately 48mV for current technologies [11].

### 3.2. Architectural Energy Optimization

The minimum functional supply voltage places a strict lower bound on the dynamic voltage scaling range in subthreshold operation. However, in this section we show that

dynamic voltage scaling is not necessarily energy efficient over this entire subthreshold voltage range. The energy per instruction can be expressed as follows:

$$E_{inst} = E_{cycle}CPI$$

where  $E_{cycle}$  is the average energy per cycle and  $CPI$  is the average number of cycles per instruction. Clearly  $CPI$  is independent of the supply voltage, but it is important when making architectural trade-offs.

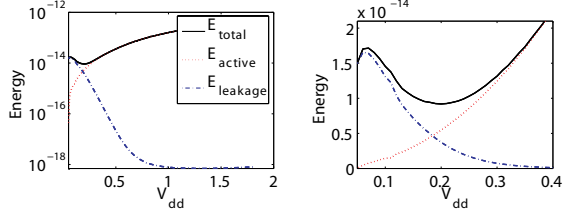
The total energy per cycle is further expressed as the sum of the dynamic energy and leakage energy, as follows:

$$E_{cycle} \propto (\alpha \frac{1}{2} C_s V_{dd}^2 + V_{dd} I_{leak} t_{clk})$$

where  $\alpha$  is the *activity factor*, which is the average number of transistor switches per transistor per cycle,  $C_s$  is the total circuit capacitance,  $V_{dd}$  is the supply voltage,  $I_{leak}$  is the leakage current, and  $t_{clk}$  is the clock cycle time.

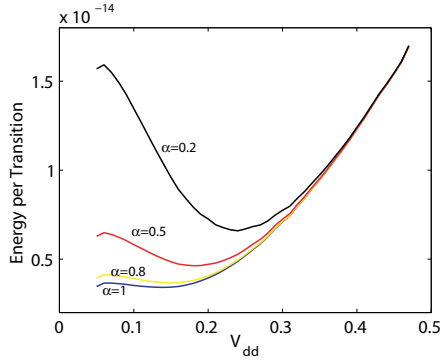
From this expression, it is clear that the dynamic energy reduces quadratically over both the superthreshold and subthreshold operating ranges. However, the behavior of the leakage energy is different in superthreshold and subthreshold operating ranges. At superthreshold supply voltages, the cycle time  $t_{clk}$  increases linearly with lowering the supply voltage while at the same time the leakage current reduces approximately linearly [17]. Hence, the leakage energy remains nearly constant. Therefore, reducing the supply voltage improves overall energy efficiency due to the reduction of dynamic energy. This is shown in Figure 3a, which shows SPICE simulation results for a 20-stage inverter chain in 0.18 $\mu$ m technology. However, at subthreshold voltage the cycle time  $t_{clk}$  increases exponentially by voltage scaling while the leakage current continues to reduce approximately linearly. Hence, the leakage energy will increase with reduced supply voltage while the dynamic energy reduces, resulting in an *energy-optimal supply voltage*, as shown in Figure 3b. Note that at the energy-optimal voltage, the leakage energy and dynamic energy are approximately balanced, making further reduction of the supply voltage energy inefficient due to the disproportionate increase in leakage energy. It can be further shown that the energy-optimal voltage is independent of the operating temperature and transistor threshold voltage because they impact the cycle time and leakage current in an opposite manner, such that their influences cancel.

The above analysis shows that a particular design has a fundamental limit to its energy efficiency, regardless of its operating frequency. The maximum energy efficiency is accomplished when the design operates at its energy-optimal voltage,  $V_{min}$ . Since a lower  $V_{min}$  results in a higher energy efficiency, it is important to determine which factors affect  $V_{min}$  and to perform architectural trade-offs that reduce  $V_{min}$  within performance constraints. A design with a



**Figure 3. Energy as a Function of Voltage. Energy for a 20-stage inverter chain over varied voltages (From [17]).**

higher ratio of dynamic-to-leakage energy will have a lower  $V_{min}$ , as the leakage energy increase will not offset the gains in dynamic energy as quickly as supply voltage is reduced. This is illustrated in Figure 4, where the energy per transition is shown for a 20-stage inverter chain as simulated for different activity factors,  $\alpha$ .



**Figure 4. Energy-Optimal Operating Points. Energy for 20-stage inverter chain for varied voltage. A minimum energy voltage exists due to increasing leakage as voltage decreases (From [17]).**

As can be seen in Figure 4,  $V_{min}$  increases as the activity factor is reduced from 1 to 0.2 transitions per cycle, because the dynamic to leakage current ratio is proportional to the activity factor:  $\frac{I_{dynamic}}{I_{leakage}} \propto \alpha$ . Similarly, the ratio of dynamic to leakage energy is inversely proportional to the cycle time, because leakage energy increases linearly with cycle time:  $\frac{E_{dynamic}}{E_{leakage}} \propto \frac{1}{t_{clk}}$ . Using our simulations and the fact that cycle time exponentially increases with a decrease in supply voltage, as previously shown, it is possible to derive the following approximate expression for  $V_{min}$ :

$$V_{min} \propto \ln\left(\frac{t_{clk}}{\alpha}\right)$$

Hence, the dependence of  $V_{min}$  on the design characteristics can be expressed using only two parameters,  $\alpha$  and  $t_{clk}$ . In our analysis, we fit the above expression to SPICE-based data for a 0.18 $\mu$ m process and verify the accuracy of the fitted expression for a number of designs.

Based on the above analysis, it is clear that architectural optimization for maximum energy efficiency is dramatically different in subthreshold and superthreshold designs. In superthreshold designs, maximum energy efficiency is obtained by reducing the total switched capacitance and by improving the operating frequency, thereby for allowing more dynamic voltage scaling. Hence, adding circuits that switch rarely but improve the cycle time or improve CPI, such as a value predictor, can aid energy efficiency. In general, increasing design complexity can improve energy efficiency as long as the total switched capacitance is not increased significantly and the cycle time or CPI is improved.

However, for subthreshold operation, additional circuitry that switches rarely and does not impact dynamic energy significantly can greatly reduce energy efficiency due to the additional leakage contributed by these additional gates. From the above analysis, it is clear that not only  $C_s$  needs to be held constant or reduced, but also  $\alpha$  must be increased for high energy efficiency. A high  $\alpha$  value corresponds to a high transistor utilization, meaning that the portion of inactive gates in a cycle is reduced. Consider two designs with an equal number of devices that are equally computationally-efficient (*i.e.* they require an identical number of switches to finish an instruction). The design with a higher  $\alpha$  is more energy-efficient for several reasons. First, a higher  $\alpha$  allows for a lower  $V_{min}$  and therefore, a lower dynamic energy. Second, because fewer devices are leaking at any given time, the leakage energy is reduced. Finally, because the average number of switches per cycles is higher, it takes less time to finish the computation, which further reduces leakage energy per instruction. Note that in this scenario, CPI is inversely proportional to  $\alpha$ . From this perspective, optimization of CPI has an increased importance in subthreshold microprocessor design, as it not only reduces leakage by eliminating idle devices but further impacts dynamic power through the reduction of  $V_{min}$ .

Hence, the optimization landscape for subthreshold operation is significantly more complex because it depends strongly on all four factors: CPI,  $C_s$ ,  $\alpha$ , and  $t_{clk}$ . Furthermore, the dependence of  $C_s$ ,  $\alpha$ , and  $t_{clk}$  on the physical implementation make it difficult to determine the subthreshold energy efficiency without studying the detailed implementation of a design. A study of energy-efficient subthreshold designs must therefore include a detailed comparison of physical implementations. We therefore present such a study in the next section.

#### 4. Architectural Trade-off Analyses

In this section, we perform a detailed trade-off study to determine which ISA and microarchitectural features work best for reducing energy at subthreshold voltages. We first examine the trade-off between instruction set expressive-

ness (which leads to compact code size) and control logic complexity (which is reduced with simpler instructions). Additionally, we examine 21 sensor network processor designs, each implemented in the IBM 0.13 $\mu\text{m}$  fabrication process.

#### 4.1. Experimental Framework

For each of the 21 processors designed for simulation, the minimum operational energy dissipation needs to be determined. The process of accurately finding the optimal operating voltage point for minimization of energy usage involves careful design and simulation of the processors and the memories with which they interface.

Upon realization of a given processor in synthesizable Verilog, the design is synthesized for optimum delay using Synopsys Design Compiler. The corresponding timing constraint is then relaxed by 30% in order to obtain a design that is more balanced in terms of area and delay than the original. Then the design is placed and routed using Cadence Sedsm, which in turn yields the wire capacitances. The design is then back annotated to get a more accurate delay profile. Next, all of the studied applications are simulated on the current design to obtain switching and CPI results, which is then used by PrimePower to compute active and leakage power.

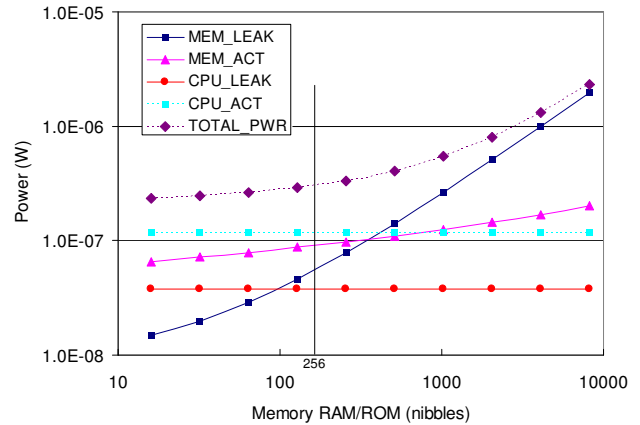
The first memory component designed to interface with the CPUs is a semi-custom, MUX-based RAM which is capable of operating in the subthreshold regime. The SRAM core is designed with structural Verilog while the decoder and MUX logic are written in behavioral Verilog and synthesized by Synopsys Design Compiler. The cells are, for the most part, placed and routed using Sedsm in order to minimize size. Subsequently, steps similar to those used with the CPU simulations are pursued to obtain dynamic and static power. The ROM, which serves as the other memory component with which the CPUs communicate, is designed using NMOS pull-down transistors to represent a logic zero. The percentage of 1's to 0's is the main factor in the determination of the leakage and short circuit power numbers. Inspection of the instruction code yields an average of 40% zeros. Power for the decoder and MUX are then determined using PrimePower, while SPICE helps determine the overall dynamic and leakage power.

With all power information at hand, SPICE simulations are created to generate fitted curves showing how frequency, as well as active and leakage power, scale with diminishing voltage. Next, total leakage and active energy per cycle for all CPU and memory designs are computed based on aforementioned SPICE-derived curves for a voltage range of 100mV to 600mV to identify the optimal-energy voltage point. Thus, the voltage at which a given design is most energy-efficient and has the least energy per cycle is deter-

mined. Finally, in order to calculate the amount of energy dissipated per instruction, the average CPI, which is determined when the applications are simulated, is used.

#### 4.2. ISA Optimizations

Instruction set design is a critical factor in the development of a sensor network processor, because the memory and ROM used to hold instructions and the control logic used to implement instructions will dissipate static and dynamic energy. In fact, memory size and control logic size form a fundamental trade-off in instruction set design for our sensor processor. With a simpler instruction set, code size will grow while control size stays small. Conversely, with a more expressive instruction set, code size decreases at the expense of more complex control logic.



**Figure 5. Logic vs. Memory Energy Trade-off. Relative contributions of energy demands due to the processor unit and the memory components, for varying memory size.**

The critical nature of memory and ROM minimization is illustrated in Figure 5. The graph shows the leakage (LEAK) and active (ACT) average power components for varied memories combined with our most energy-efficient sensor network processor. The memory architectures are composed of 1/2 RAM and 1/2 ROM, and the results shown are averages across the entire sensor network processing benchmark set.

As memory demands increase, overall energy consumption shifts to leakage in the memory arrays. We make two observations from this study. First, memory demand, especially that imposed by instructions, must be reduced. Hence, we aggressively pursue a dense instruction set encoding. Second, it is critical to reduce memory cell leakage. While we do not address memory cell leakage in this paper, we are pursuing novel memory architectures that reduce leakage through reduced transistor counts and additional volt-

age scaling opportunities. We briefly mention these experiments in Section 5. They will be fully detailed in a future report.

Mnemonic	Operation	Length nibbles
ADD	Performs addition	2 or 3
SUB	Performs subtraction	2 or 3
AND	Performs logical and	2 or 3
OR	Performs logical or	2 or 3
XOR	Performs logical exclusive or	2 or 3
SHFT	Shifts the accumulator	2 or 3
LOAD	Loads the accumulator	2 or 3
STOR	Stores the accumulator	2 or 3
DW_BK	Sets BLCK and DW specifiers	2
PTR_INC	Increments pointer register	2
PTR_DEC	Decrements pointer register	2
PTR_LOAD	Loads acc. with pointer reg.	2
PTR_STOR	Stores acc. into pointer reg.	2
CALL	Calls a function	3
RET	Returns from a function	1
JUMP	Conditionally jumps to target	4
NOP	No operation	1

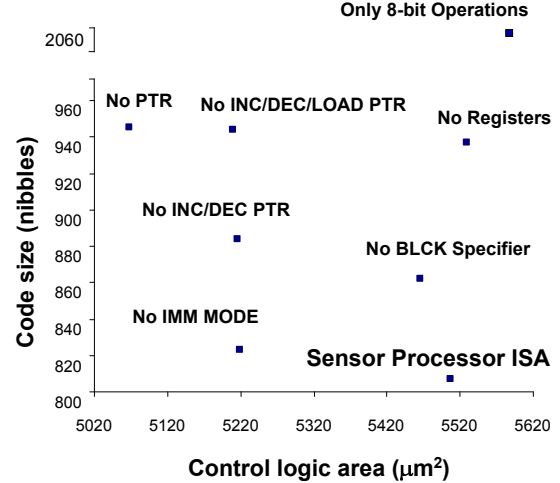
**Table 3. Sensor Network Processor Instruction Set Summary. Listed is the sensor processor ISA implemented for all of the studied designs.**

Table 3 summarizes our sensor network processor instruction set. The table lists the instruction mnemonic, a short description of the instruction, and its size in nibbles. Our instruction set is a simple 32/16/8-bit single-operand ISA. The instruction set contains two register banks: a 4-entry 32-bit integer register file and a 4-entry 16-bit pointer register file. The pointer registers hold memory addresses, thus the architecture can address up to 64 kBytes of storage. All computational instructions are of the form:

$$(Acc) \leftarrow (Acc) \otimes operand$$

where *operand* is either i) a general-purpose register operand, ii) a pointer register which specifies a value in memory, iii) a direct 6-bit memory address, or iv) a 2-bit signed immediate value.

Figure 6 illustrates the impact of a number of ISA optimizations we implement to reduce code size, at the expense of increased control complexity. The *PTR* instructions provide efficient memory addressing by providing a compact means, in the form of pointer registers, to express addresses and efficiently implement strided accesses. Eliminating the pointer registers, while reducing control complexity, has a significant impact on code size, increasing overall code size by 16%. Eliminating the general-purpose registers has



**Figure 6. Impact of ISA Optimization on Code Size and Control Logic Complexity. Trade-off between ISA expressiveness (which results in smaller code size) and control logic size.**

a similar effect on code size, with little benefit to control complexity. The *DW\_BK* instruction sets both *BLCK* and *DW* specifiers. The *BLCK* specifier is used to take advantage of locality in absence of caches, where one can choose the working block in memory and therefore reduce the number of address bits in order to shorten instruction. Eliminating the block specifier increases code size about 6% with a slight increase in control complexity. Finally, eliminating the ability to process 16- and 32-bit data types (implemented via the *DW* specifier, which determines the virtual width of the datapath) bloats code size by nearly 2.5x. This increase is due to the many additional instructions required to implement 16- and 32-bit operations (e.g., a 16-bit operation requires an 8-bit add, plus an 8-bit add-with-carry.) Removing support for multiple data widths provides little benefit to control complexity.

### 4.3. Microarchitectural Design Space Analysis

Figure 7 illustrates our sensor network processor microarchitecture. The figure shows the most comprehensive microarchitecture studied. Many of the variants only include a subset of the features shown in the figure.

The processor contains three pipeline stages. The *IF-STAGE* contains instruction memory and ROM, and a prefetch buffer. The prefetch buffer is a 32-bit buffer containing up to four instructions. It is filled from instruction memory whenever the decoder finds that it does not contain a complete instruction. The *ID-STAGE* contains the register file, which is a 4-entry 32-bit register file. Values from the register file are sent to the accumulator, which is a 32-bit register. The accumulator is the only place that instruction results are stored. Option-

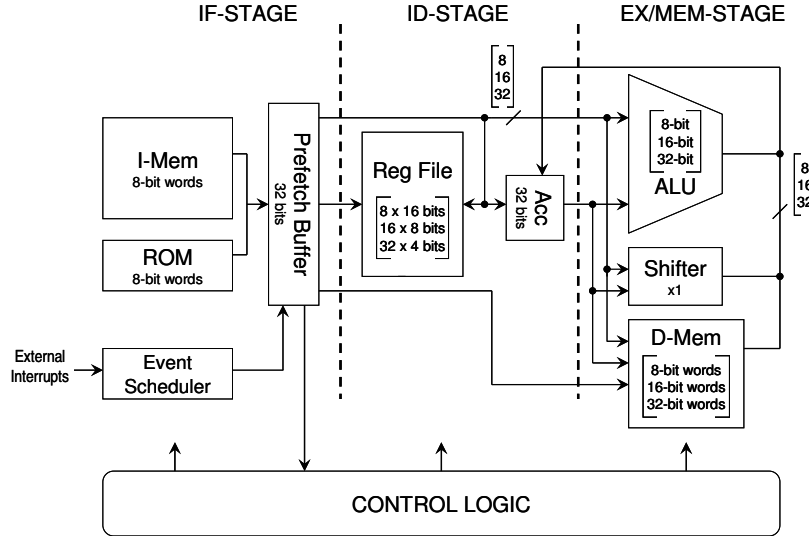


Figure 7. Sensor Network Processor Microarchitecture Overview.

ally, a datapath exists between the accumulator and the register file, which allows accumulator values to be written back to the register file. The *EX-STAGE* contains the functional units and data memory.

External events, *e.g.*, from sensors, are processed by the event scheduler. The scheduler has two event inputs, which permit high-priority and low-priority events. Low-priority events are handled in the order that they arrive to the sensor network processor. High-priority events, on the other hand, are also processed in order, but they may preempt the processing of a low-priority event. When a low-priority event is preempted, the sensor processor operates with a separate set of registers and internal control state bits. Thus, once the high-priority event is finished processing execution can resume undisturbed for the preempted low-priority event.

The scheduler is the extreme case of code density vs. control size. A software-only version of the scheduler is 224 nibbles in size (including shared data and instructions). Considering  $8\mu\text{m}^2$  per bit, the memory size to hold the scheduler is  $7868\mu\text{m}^2$ , while the hardware scheduler has relatively modest area requirements of only  $3147\mu\text{m}^2$ .

Figure 8 shows the performance and energy of 21 physical designs. The designs are labeled to indicate: i) the number of pipeline stages (1s, 2s, or 3s), ii) the number of memories (v - one memory, h - I and D memory), iii) the datapath width (8w, 16w, or 32w), and iv) the existence ( $\_r$ ) of explicit registers (designs without explicit registers store register values in memory). In the figure, designs closer to the origin are faster and more energy-efficient than designs further away. The designs on the pareto-optimal curve represent the best designs developed, with varying energy and performance trade-offs. Designs off of the pareto-optimal curve are not worth implementing because at least one of the designs on the curve is both faster and more energy-efficient. As shown in Figure 8, the designs on the pareto-

optimal curve are compromising designs, in that they are not fully pipelined or maximal width at the same time. This represents the careful balance that designs must make at subthreshold voltage levels to be at the same time CPI-efficient, area-frugal, and with high transistor utility. For each design on the pareto-optimal curve, we show the area (in  $10^4\mu\text{m}^2$ ), the activity factor (in  $10^{-1}$  transitions per transistor per cycle), and the CPI.

Also highlighted in the pareto-optimal curve are a few representative non-winning designs. The *2s\_v\_32* design takes too large of a CPI degradation (due to a unified memory) to remain an optimal design. The *3s\_h\_08w* design has a large area increase due to pipelining, along with a commensurate decrease in activity rate, resulting in a non-optimal result. Design *2s\_h\_08w* suffers a similar fate.

Figure 9 highlights how architectural energy optimization changes in the subthreshold voltage domain. Two designs are shown in the graph, a 32-bit 2-stage design and a 16-bit 3-stage design. At 1.2V the two designs have roughly the same energy demand per cycle, but the 32-bit design has a much lower CPI, resulting in a more energy-optimal design at 1.2V. In addition, at superthreshold voltages the 32-bit 2-stage design also reduces overall activity due to wider datapaths. At subthreshold voltages, the tables are turned. The 16-bit 3-stage design has both higher transistor utility and smaller area, yielding a lower  $V_{min}$  and a much greater energy efficiency.

## 5. Insights and Future Designs

In this paper, we examined the landscape of energy optimization for sensor processors. We observed that sensor network processors, while having very tight energy constraints due to their small form factors, have very low performance demands for a wide variety of sensor applications.

## Energy vs. Performance

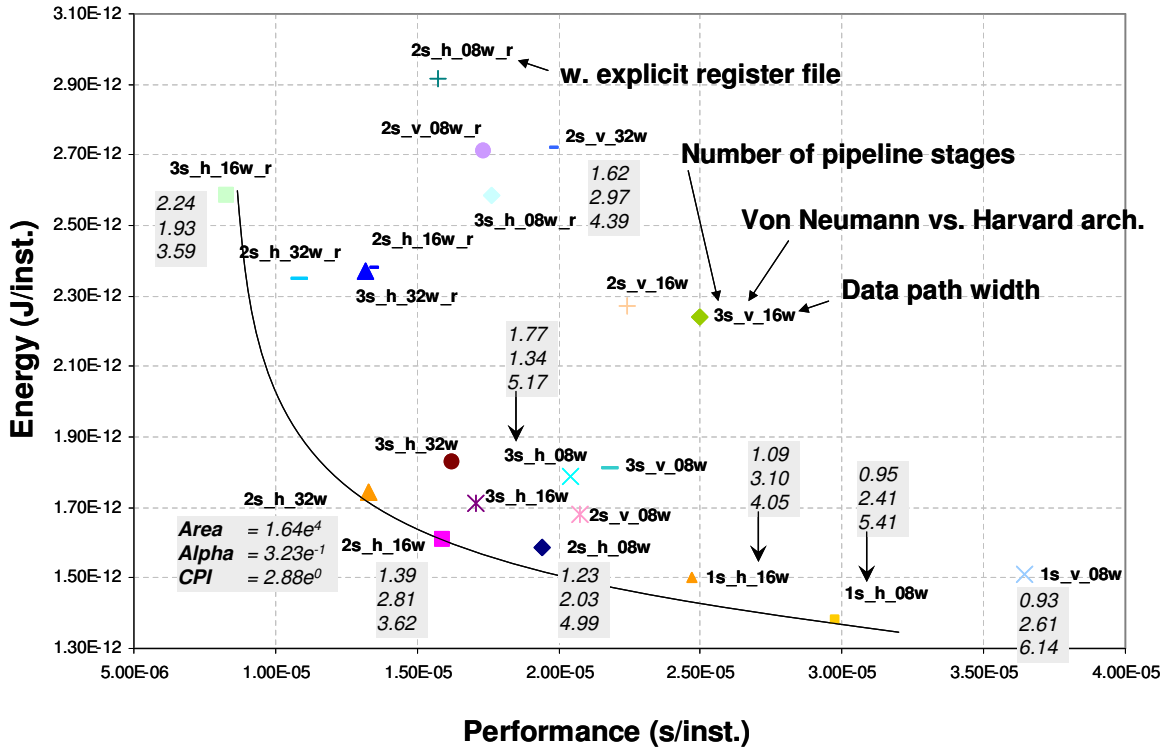


Figure 8. Processor Energy vs. Performance. Pareto chart for relative performance vs. energy demand. The designs on the curve are pareto-optimal designs.

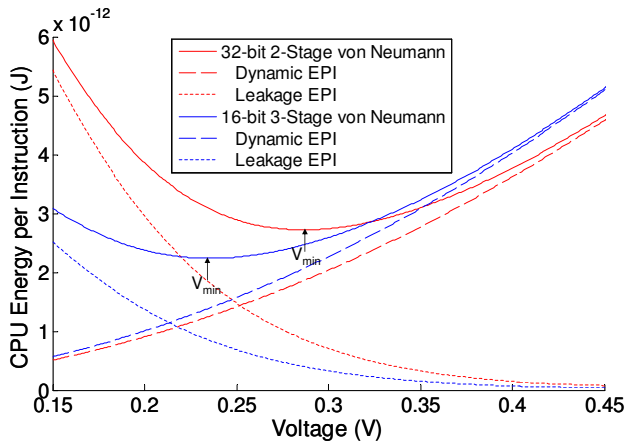


Figure 9. Energy Demand vs. Voltage

We reviewed the basic precepts of subthreshold circuit operation, namely, that when voltage drops below the threshold voltage, circuit evaluation delay grows exponentially. We also highlighted a corollary to the exponential delay growth in that subthreshold designs exhibit an optimal minimal voltage, above which dynamic energy consumption dominates, and below which excessive run times lead to increased static energy demands. In conjunction with this review of subthreshold design, we

introduced the basic tenets of microarchitectural energy optimization at subthreshold voltages. Specifically, energy optimal subthreshold-voltage sensor network processors strike a careful balance that i) reduces overall area, ii) increases the utility of transistors, and iii) maintains acceptable CPI efficiencies.

To confirm these precepts of energy-efficient subthreshold-voltage design, we examined 21 sensor network processor designs. Each design was implemented in an IBM  $0.13\mu\text{m}$  fabrication process and analyzed using a commercial VLSI design flow. We found that compromising designs that strike a careful balance between the competing factors of CPI, transistor utilization, and area led to the overall lowest-energy designs. Our most energy-efficient design is a simple sensor network processor, with a ROM/SRAM memory combination, 8-bit datapath, and a compact ISA design. The design operates at 235mV with a clock frequency of 182kHz. Even at this deep subthreshold voltage, the design is still able to run 4.1 times faster than necessary to meet the worst-case computational demands of our mid-bandwidth sensor processor application set. Moreover, this design consumes nearly an order of magnitude less energy than previously published sensor processor designs. If coupled with a 2g

vanadium oxide battery, containing 720 mA-hr of energy, the design would be able to run non-stop for more than 25 years!

Additionally, we examined the trade-offs between instruction set expressiveness (which leads to compact code size) and control logic complexity (which is reduced with simpler instructions). We found that the decreases in code size always outweighed the increases in control logic size, even for simple programs. Thus, compact ISA designs are quite appropriate as they decrease memory demands.

In an effort to validate the results, we implemented a prototype subthreshold-voltage sensor network processor test chip. The test chip is 2.5mm x 2.5mm, and it contains 6 sensor processor and memory pairs, 4 additional experimental memories, 4 bulk-silicon solar cells, and a test harness. The sensor network processors are implemented with a variety of standard cell designs, ranging from low- $V_{th}$  commercial cells to high- $V_{th}$  experimental cells optimized for subthreshold operation. The test memories range from standard memories, implemented with MUX-combined SRAM arrays to experimental memories with 4- and 3-transistor low-leakage SRAM cells. The bulk-silicon solar cells are PMOS devices that, when excited with sunlight, produce a nominal 180mV power source. Through simulation, we have estimated that indoor lighting applied to the solar cells will produce enough energy to power subthreshold-voltage logic with approximately 1/3 the area of the solar cell. The largest solar cell is intended to produce enough current that it can be measured off-chip. Finally, the test harness provides a SCAN interface between the outside world and all state (memory and registers) contained on the test chip. In addition, the SCAN interface can be used to reset and restart any processor or test harness contained on the test chip. This chip has been fabricated by IBM and is currently being tested. We will report on the results of this test chip in future publications.

Looking forward, there is certainly an opportunity to better the energy-efficient designs presented in this paper. There exists headroom to further lower  $V_{dd}$ , as all of the designs examined in this study have much more performance capability than that required by our mid-bandwidth sensor network processor application set. Currently, we are examining additional microarchitectural optimizations to further improve CPI while mitigating area and activity degradation.

If we only concentrate on the low-bandwidth sensor applications, there is much more additional headroom to discover designs with significantly lower minimal  $V_{dd}$  operating points. However, to find these designs, they will certainly have to provide extremely high transistor utilization, while at the same time being small with efficient CPIs. We are currently exploring the possibility of reconfigurable microarchitectures as a means to explore this deep subthreshold voltage domain. Additional challenges are found here

as process variation becomes quite a significant design factor below 150 mV [17].

**Acknowledgments.** We would like to thank Valeria Bertacco for her input to this paper. This work is supported by grants from NSF and the Gigascale Systems Research Center.

## References

- [1] ARM, Ltd. website. <http://www.arm.com>.
- [2] Monitoring in anaesthesia. In [http://www.liv.ac.uk/~afgt/Phys\\_Meas\\_Notes.pdf](http://www.liv.ac.uk/~afgt/Phys_Meas_Notes.pdf), 2004.
- [3] Multilog, multilogpro and ecolog weather stations. [www.fourier-sys.com/pdfs/data\\_loggers/](http://www.fourier-sys.com/pdfs/data_loggers/), 2004.
- [4] Online resource for information on data compression. <http://www.data-compression.info/Algorithms/RLE>, 2004.
- [5] C.-Y. Chong. Sensor networks: Evolution, opportunities and challenges. In *Proc. of the IEEE, Volume:91, Issue:8*, 2003.
- [6] V. Ekanayake, C. Kelly, and R. Manohar. An ultra low-power processor for sensor networks. In *Proc. International Conference on Architectural Support for Programming Languages and OS*, 2004.
- [7] J. Feng, F. Koushanfar, and M. Potkonjak. System-architectures for sensor networks issues, alternatives, and directions. In *Proc. International Conference on Computer Design*, 2002.
- [8] J. L. Hill. System architecture for wireless sensor networks. In *University of California, Berkeley*, 2003.
- [9] J. Kao, S. Narendra, and A. Chandrakasan. Subthreshold leakage modeling and reduction techniques. In *Proc. International Conference on Computer-Aided Design*, Nov. 2002.
- [10] F. Koushanfar, V. Prabhu, M. Potkonjak, and J. Rabaey. Processors for mobile applications. In *Proc. of IEEE International Conference on Computer Design*, 2000.
- [11] J. D. Meindl and J. A. Davis. The fundamental limit on binary switching energy for terascale integration (TSI). In *IEEE JSSCC vol. 35*, Feb. 2002.
- [12] J. Rabaey, J. Ammer, T. Karalar, B. O. S. Li, M. Sheets, and T. Tuan. Picoradios for wireless sensor networks: The next challenge in ultra-low-power design. In *Proc. International Solid-State Circuits Conference*, Feb. 2002.
- [13] C. Schurgers and M. B. Srivastava. Energy efficient routing in wireless sensor networks. *MILCOM*, Oct. 2001.
- [14] A. Wang and A. Chandrakasan. A 180mv fft processor using subthreshold circuits techniques. In *International Solid-State Circuits Conference ISSCC*, 2004.
- [15] B. A. Warneke and K. S. Pister. An ultra-low energy micro-controller for smart dust wireless sensor networks. In *Proc. International Solid-State Circuits Conference*, 2004.
- [16] D. J. Wheeler and R. M. Needham. TEA, a tiny encryption algorithm. *Lecture Notes in Computer Science*, 1008, 1995.
- [17] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner. Theoretical and practical limits of dynamic voltage scaling. In *Proc. Design Automation Conference*, 2004.