INVITED PAPER

Flexible Circuits and Architectures for Ultralow Power

A programmable gate array optimized for ultralow-power operation may provide hardware flexibility and allow rapid low-cost implementation of many new applications.

By Benton Highsmith Calhoun, *Member IEEE*, Joseph F. Ryan, Sudhanshu Khanna, Mateja Putic, *Member IEEE*, and John Lach, *Senior Member IEEE*

ABSTRACT | Subthreshold digital circuits minimize energy per operation and are thus ideal for ultralow-power (ULP) applications with low performance requirements. However, a large range of ULP applications continue to face performance constraints at certain times that exceed the capabilities of subthreshold operation. In this paper, we give two different examples to show that designing flexibility into ULP systems across the architecture and circuit levels can meet both the ULP requirements and the performance demands. Specifically, we first present a method that expands on ultradynamic voltage scaling (UDVS) to combine multiple supply voltages with component level power switches to provide more efficient operation at any energy-delay point and low overhead switching between points. This system supports operation across the space from maximum performance, when necessary, to minimum energy, when possible. It thus combines the benefits of single-V_{DD}, multi-V_{DD}, and dynamic voltage scaling (DVS) while improving on them all. Second, we propose that reconfigurable subthreshold circuits can increase applicability for ULP embedded systems. Since ULP devices conventionally require custom circuit design but the manufacturing volume for many ULP applications is low, a subthreshold field programmable gate array (FPGA) offers a cost-effective custom solution with hardware flexibility that makes it applicable across a wide range of applications. We describe the design of a subthreshold FPGA to support ULP operation and identify key challenges to this effort.

KEYWORDS | Dynamic voltage scaling; energy scalability; panoptic DVS; PDVS; reconfigurable logic; subthreshold; subthreshold FPGA; UDVS; ultra DVS; ultralow power

I. INTRODUCTION

In this paper, we examine methods to incorporate flexibility into the circuits and architectures of ultralow-power (ULP) systems. We specifically address flexibility in the achievable performance of ULP systems (e.g., by expanding on DVS) and flexibility in the type of operations that ULP systems perform (e.g., by designing a subthreshold FPGA). Inflexible designs are appropriate for some ULP scenarios, but we will show that flexibility greatly expands the space of applications where ULP systems practically can apply. Since flexibility inherently opposes energy efficiency, we propose methods that minimize the overhead of incorporating flexibility while maximizing the benefits of that flexibility.

The rapid shrinking of transistor sizes according to Moore's Law [1] has propelled integrated circuits (ICs) into a power-limited era [2]. For digital circuits, a fundamental trade-off exists between maximum circuit speed and circuit power consumption. Faster circuits must consume more power. This trade-off constrains circuit design for applications across a broad performance range. Power consumption limits high-performance circuits (e.g., processors in servers or desktops) because of their large power density that leads to potentially damaging high temperatures on the chip. Embedded applications (e.g.,

Manuscript received April 18, 2009; revised August 10, 2009. Current version published January 20, 2010. Funding for this work came in part from a DARPA Young Faculty Award, SCEEE, UVA FEST, and NSF.
The authors are with the Charles L Brown Department of Electrical and Computer Engineering, University of Virginia, Charlottesville VA 22904-4743 USA (e-mail: bcalhoun@virginia.edu).
Digital Object Identifier: 10.1109/JPROC.2009.2037211

cell phones, portable electronics) often get by with slower operation, but they face power constraints primarily from the need to extend battery lifetimes by reducing the amount of energy consumed to perform each operation. Recently, a new class of ultralow-power (ULP) applications has appeared that seeks to expand the conventional design space toward miniaturized, embedded, and very long lasting operation.

Ultralow-power circuits derive their name from the stringent limits on power consumption that they must maintain. Understanding the types of applications that receive the ULP label will help to identify the source of the tight power constraints. Examples of ULP applications include distributed wireless sensor nodes, medical implants, biotelemetric sensors, microvehicles, and RFID tags. All of these applications have a small form factor, which limits the amount of energy storage available. They also tend to require extended operational lifetimes in the range of weeks to months. Successful operation over a long period of time with a limited amount of energy demands very low energy consumption by the circuits.

Two features of ULP applications that we can leverage to reduce power consumption are that they tend to have relatively low speed requirements and that they tend to be fairly specific (e.g., do just one thing or one small set of things). We can leverage slow operation to lower energy use according to the trade-off between energy and performance. We also can leverage specific operation to trim unnecessary components from the circuit to streamline power consumption. Both of these methods commonly appear in ULP research.

An approach called subthreshold circuit operation uses low supply voltages to minimize the energy consumed by digital circuits and has thus become a popular option for research in ULP applications. Section II describes subthreshold circuits in detail. One major limitation of subthreshold operation is decreased performance. Some ULP applications can get by with only slow operation, but there are many scenarios in which slow operation becomes intolerable. Miniature wireless sensor nodes provide one example of this. The amount of processing and the rate of processing required on the sensor for looking at incoming data can be quite low, making a perfect scenario for subthreshold operation to save power. However, if the sensor detects a critical event (e.g., data point requiring more scrutiny, life critical health event, etc.), the system may need to increase the amount of signal processing or increase the processing speed to properly handle the event. The increased processing and speed may exceed the capabilities of subthreshold circuits. This means that existing approaches to subthreshold circuit design cannot apply to systems that require periods of higher performance operation.

Many ULP systems also are very specialized. For example, an implanted neural recording device will implement signal processing for neural spike detection and sorting. The energy efficient solution for this application would be an application-specific integrated circuit (ASIC) implementation, but the targeted nature of this hardware makes it impossible to reuse in a different ULP application.

While consistently slow operation and special purpose hardware save power, they also limit the scope and applicability of ULP circuits. In the high-performance end of the design space, FPGAs offer efficient hardware-based operation along with the flexibility to reconfigure that hardware to perform different tasks. We observe that the excellent trade-off between efficiency and flexibility that FPGAs provide in higher performance circuits could dramatically expand the applicability and scope of ULP circuits. To make this work, the FPGA concept requires a redesign to enable low voltage operation and to focus on energy reduction along with performance.

This paper suggests circuit and architecture approaches that allow ULP systems to adjust their performance to higher speeds when necessary and to reconfigure their functionality when required. This increased flexibility broadens the applicability of subthreshold circuits and improves their energy efficiency across this expanded design space. Section II places this discussion in context by covering the basics of subthreshold operation and briefly surveying the state of the art in subthreshold designs for ULP applications. Section III explores methods to trade off circuit speed with energy consumption and describes our suggested approach for integrating ULP circuits into an architecture that supports energy/performance trade-offs across a broad range. Section IV considers how hardware flexibility affects energy efficiency and makes a case for a subthreshold FPGA designed for ULP systems. Section V concludes the paper.

II. SUBTHRESHOLD OPERATION

Subthreshold circuits apply a supply voltage, V_{DD}, that is below the threshold voltage, V_T, of the transistors to decrease the power and energy consumption of digital circuits. The transistors are "off" by conventional definitions, but subthreshold conduction allows them to continue to operate correctly, although much more slowly due to the lower current. The most effective method to lower power for slower speed applications is to lower V_{DD}. The power consumed in a digital circuit as it switches the voltage stored on a total capacitance, C_{eff}, varies quadratically with V_{DD} according to (1), where f is the operating frequency.

$$P_{switching} = f C_{eff} V_{DD}^2 \tag{1}$$

Transistors in modern CMOS process technologies also consume leakage power, which results from leakage current that flows in the transistor when it is off. This is analogous to a leaky faucet. Leakage currents decrease

sharply with V_{DD}. Fig. 1 shows how switching power and leakage power decrease with V_{DD}. This figure shows that the lowest power consumption for a circuit will result at the lowest functional V_{DD}. Low *power* operation can be important for applications like RFID, which receives wireless power delivery. Power is the average energy consumed over time, and the lowest power setting might not always provide the lowest *energy* solution, which is more critical for battery powered systems. For example, performing an addition at a higher power setting might consume less energy if the operation takes much longer to complete at the lower power setting. In that case, the lower power would integrate over the longer delay to result in a larger energy than the energy consumed by the shorter high power computation. To find the voltage that minimizes energy per operation, we must examine the impact of voltage on delay.

Fig. 2 shows how delay through a static CMOS inverter varies with supply voltage. The delay begins to increase exponentially when the supply voltage drops below the threshold voltage of the transistors ($V_T \sim 0.4$ V). We can use the delay of a complete operation, T_D, to compute the total energy of a digital circuit, according to (2).

$$\begin{aligned} E_{\text{operation}} &= E_{\text{switching}} + E_{\text{leakage}} \\ &= T_D * (P_{\text{switching}} + P_{\text{leakage}}) \end{aligned} \tag{2}$$

Fig. 3 shows the plot of the total energy consumed in a long inverter chain. Switching energy decreases quadratically (1) and leakage energy increases due to the longer delays, resulting in a minimum energy point that typically occurs in the subthreshold region [3], [4], [36]. The primary advantage of subthreshold operation is a reduction in energy consumption of over an order of magnitude relative to operating at conventional V_{DD} (1.0 V to 1.2 V for 45 nm to 90 nm).

The primary disadvantages to subthreshold operation are reduced on-current to off-current ratio (I_{on}/I_{off}), slower speeds, and increased sensitivity to variations. The

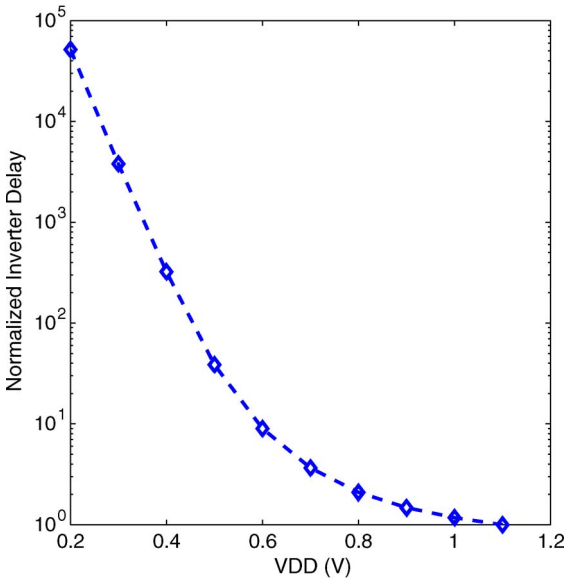


Fig. 2. *Delay of an inverter increases exponentially when V_{DD} drops below V_T.*

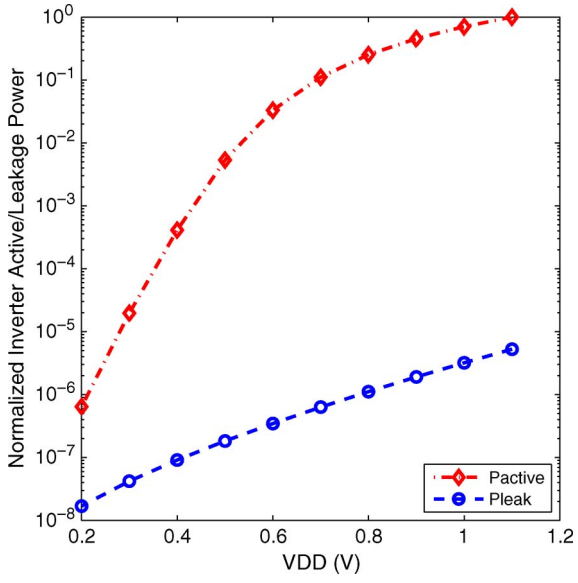


Fig. 1. *Active power and leakage power versus V_{DD}. Minimum power consumption occurs at the minimum operational V_{DD}.*

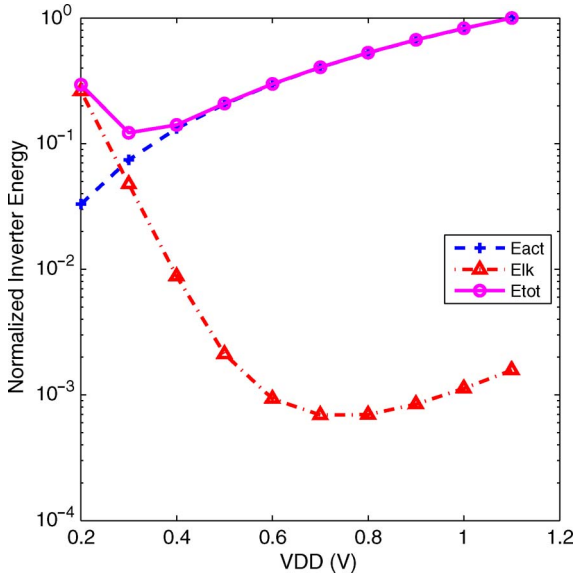


Fig. 3. *Active energy decreases with V_{DD}, but leakage energy increases, creating a minimum energy point.*

on-current of transistors in subthreshold reduces by potentially many orders of magnitude relative to strong inversion operation due to the lower V_{GS}. This results in a lower ratio of I_{on} to I_{off} that can potentially lead to functionality problems. The lower I_{on} also reduces the speed of circuits operating in subthreshold, making this region of operation inappropriate for high speed applications. The attainable speed depends heavily on the circuit topology, process technology, and operating voltage, but a good assumption is that subthreshold circuits can provide operation up to a few tens of megahertz. Finally, the threshold voltage of a transistor exhibits a normal distribution resulting from process variations, and the standard deviation of this distribution increases with process scaling to smaller technologies [27]. While threshold voltage variation affects all circuits, subthreshold operation increases the sensitivity of circuits to variations in the threshold voltage. In the subthreshold region, current depends exponentially on $V_{DD} - V_T$, leading to exponential changes in both on-current and off-current from this variation. The most significant challenge that results from variation for circuits operating in subthreshold is a further degradation of the already reduced I_{on}/I_{off} ratio. Variations in the threshold voltage can make the off-current increase and the on-current decrease, leading to circuits that cannot function.

The degraded current ratio provides a clue about which circuits are likely to fail first at low voltage. Any circuits that use a topology that reduces I_{on} relative to I_{off} will likely fail earlier than conventional static CMOS logic [34]. For example, ratioed circuits that use sizing ratios to ensure proper functionality present a problem. In subthreshold, the exponential impact of V_T variations far exceeds the linear impact of transistor size on current, so ratioed circuits (e.g., 6 T SRAM cell) should be avoided. Sizing is a weak knob in subthreshold. Other topologies that present the same problem include large transistor stacks or many parallel leakage paths. For example, an SRAM bitline during a read operation has one cell driving the bitline with its on-current while every unaccessed cell results in a leakage current path that can fight against that on-current. The aggregate leakage current can dramatically decrease the I_{on}/I_{off} ratio, leading to the inability to detect the correct value inside the accessed cell.

Previous work has addressed these challenges using circuit and architecture techniques to build chips demonstrating logic [4], [5], [45], memory [6], [41]–[43], and microcontrollers [37]–[39]. Here, we mention two key techniques that allow these subthreshold circuits to work by compensating sufficiently for the large impact of process variation. First, subthreshold designs avoid using circuit topologies that decrease the I_{on}/I_{off} ratio. Simple static CMOS gates with short stacks (e.g., less than four series transistors) provide robust operation in most cases. Subthreshold SRAM tends to use alternative bitcell topologies to improve noise margin relative to the ratioed 6T cell. For example, variation in the read static noise margin makes the 6T cell essentially unreadable in subthreshold without inducing read upsets. Alternative designs for the bitcell use a buffer to protect the storage node in the cross-coupled inverters of the cell, thereby removing the read stability concern. Second, subthreshold designs employ stronger circuit knobs than size to affect circuit behavior. In subthreshold, the exponential dependence of current on V_{GS} makes voltage a powerful knob. For example, some subthreshold SRAM designs use peripheral voltages to assist in writing (e.g., by boosting the wordline voltage) [6] or reading (e.g., by reducing V_{GS} of unaccessed bitcell passgates to lower leakage power [41]). The supply voltage itself is a very powerful knob to oppose variation. Raising V_{DD} slightly can dramatically improve robustness with only minimal impact on energy (e.g., [7]).

Subthreshold circuits provide a promising solution for ULP applications with strictly low performance solutions. However, their incompatibility with higher performance operation limits their use in a broad range of applications. The next section examines methods for expanding the operating range of ULP circuits while maintaining the ULP benefits of subthreshold operation.

III. ENERGY/PERFORMANCE SCALABILITY

The fundamental trade-off between power and speed in CMOS circuits requires a specific minimum amount of power to achieve a given speed or a maximum speed to achieve a given power. Most designs that purport to provide both low power and high performance really just optimize the circuits to reach the pareto-optimal limit of this trade-off. For a fixed high speed requirement, consuming less power than this limit is fundamentally unachievable.

Fortunately, a huge range of systems do not always run at a fixed high speed. This is especially true for embedded real-time ULP systems like unattended ground sensors, wireless microsensors, video processors, and medical monitors. These devices each require high-performance computation at (relatively rare) times, but their operating environments result in much longer durations of relaxed performance requirements. For example, an intrusion detection microsensor may run in a low energy monitoring mode for an extended period until activity is detected, and a high-performance mode is then required to rapidly process the incoming data for classification. Even during that high-performance time, saving energy is crucial for the ULP system.

We can take advantage of the bursty nature of such applications to potentially achieve both low power (e.g., low overall system power) and high performance (e.g., successful completion of all tasks on time, including tasks requiring high speed), but the challenge is to have the flexibility to operate in multiple modes while achieving this optimal efficiency both within each mode and switching between modes. To meet this challenge would require

operating at a different point along the pareto-optimal curve for each specific processing rate requirement. Dynamic voltage scaling (DVS), which reduces V_{DD} at run-time to lower power quadratically during periods of relaxed speed constraints, offers one successful approach to this problem [10], [11]. DVS maximizes energy savings when the processing rate matches the incoming workload such that we take the maximum amount of time available to process that workload. However, DVS traditionally limits voltage scaling to above-threshold voltages, preventing dynamic transitions into ULP modes. Ultra-DVS (UDVS) [9] addresses this limitation by allowing voltage hopping into subthreshold operation. Due to the large increase in delay at very low voltages (see Fig. 2), DVS can only scale into the subthreshold region when processing rate requirements are orders of magnitude lower that the maximum rate. When this is possible, however, UDVS offers substantial energy savings. At the high-performance end of the range, varying processing rates provide an opportunity to save energy.

While DVS provides the strategy for energy/performance scalability, it is typically applied globally at the chip level, with the entire chip functioning at the same V_{DD} and clock frequency. As a result, global DVS cannot provide any energy savings if the workload to a number of blocks on the chip reduces while the workload to a critical block stays the same; the entire chip must remain at the highest voltage to sustain the performance requirement of the one internal block. Some ICs have incorporated multiple DVS islands on a chip at a coarse granularity, each with its own supply rail and clock (e.g., [12], [13]), but the time and energy overhead of changing the voltage on a rail can be large. This increases the amount of time that a system must spend in a low energy mode to make up for the overhead of switching between modes, which is called the "break even time." While modern regulators can provide precise voltage values (e.g., mV accuracy), transitioning from one voltage to another using a dc–dc converter take tens to hundreds of microseconds [10], [14], [15]. This long time overhead prevents conventional DVS systems from keeping up with rapid changes in workload, thus limiting the number of processing rates such a system can achieve. The system can only change its voltage when the workload of a DVS island changes for a timescale of many microseconds. While global DVS allows for flexible energy/performance scalability, the magnitude of energy and delay overheads of global supply switching mean that the frequency of V_{DD} changes must remain low so that power savings from long periods spent at the lower voltage offset the overhead.

We present new energy/performance scalability techniques that expand on [9] to: 1) pursue the energy-efficiency limits within both high performance and ULP modes, and 2) provide rapid and low energy transitions between these modes, thus providing a minimum break even time.

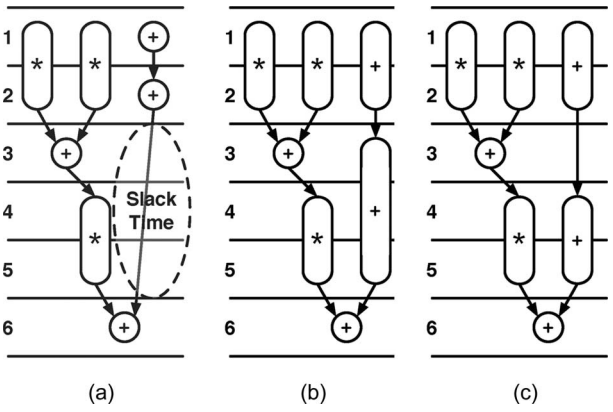


Fig. 4. *Slack fill for energy minimization given a performance requirement. (a) Adder idles during slack in c-steps 3–5. (b) Minimum dynamic energy schedule with multiple V_{DD}s. (c) Alternate schedule using fewer components.*

First we consider the efficiency of a system while operating within a given mode (e.g., near the high end of the performance range), where the goal is to meet a task's performance requirement with the smallest possible energy. Consider the dataflow graph (DFG) in Fig. 4(a) implemented on a synchronous dataflow architecture, where each control step (c-step) corresponds to one clock cycle. Assuming a multiplication and addition at the highest source voltage V_{DDH} take two and one c-steps, respectively, the DFG can be executed in a minimum of six c-steps. However, the two additions not on the DFG's critical path could be executed at lower voltages and consume less energy per operation without affecting the DFG latency.

Most modern ICs include several different supply voltages (e.g., separate supplies for I/O, for core logic, and for SRAM), and architectures using multiple supply voltages (multi-V_{DD}) to lower power have become commonplace (e.g., [16], [17]). Multi-V_{DD} designs leverage latency slack like we observed in Fig. 4 and could achieve the minimum dynamic energy schedule shown in Fig. 4(b). This implementation requires three V_{DD}s and three adders (the V_{DDH} adder used in c-step 3 can be reused in c-step 6). Fig. 4(c) shows an alternate schedule that sacrifices some dynamic energy relative to the optimal schedule but can be implemented with only two V_{DD}s and two adders. While multi-V_{DD} designs do require additional consideration for routing multiple supplies and for level converting circuits between voltage islands, the savings outweigh the area overhead [16]. For example, an implementation of a processor in [17] accomplishes a multi-V_{DD} implementation without any area overhead after revisiting the physical design. Two significant limitations of multi-V_{DD} architectures are: 1) each component is permanently assigned to a single voltage rail, thus limiting flexibility, and 2) the significant time and energy required to change the voltage

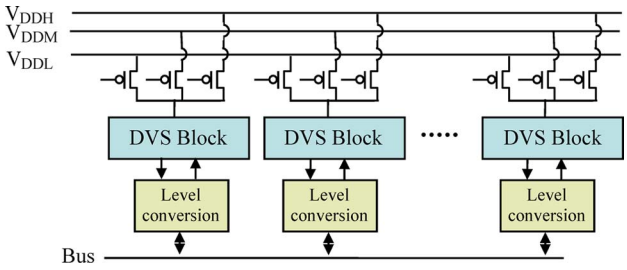


Fig. 5. ***Example PDVS system with multiple blocks and three voltage levels.***

on the rails using the dc–dc converter, thus suffering from the same energy/performance scalability issues as traditional global DVS.

We introduce a new architecture shown in Fig. 5, Panoptic DVS (PDVS), that uses header switches that enable each component to switch between several available supply voltages, enabling a single arithmetic component to implement multiple sequential operations within a single algorithm iteration at various processing rates [19]. Adding extra header switches only marginally increases total leakage (e.g., by a few percent) due to reduced V_{DS} across those headers, and conventional leakage reduction techniques like reverse body bias can further decrease this overhead if desired. With this fine spatial voltage control granularity, PDVS can achieve the same schedules as multi-V_{DD}, but often with fewer components, since a single component can operate with different energy/delay characteristics at different times by switching between voltage rails. In addition, the header switches provide fine temporal voltage control granularity, as components can quickly switch processing rates for both intra-DFG changes (improving efficiency within both high performance and ULP modes) and coarser-grained changes in workload (providing rapid and low energy transitions between modes).

Consider again the DFGs in Fig. 4. Existing single-V_{DD} and multi-V_{DD} high-level synthesis algorithms can determine minimum energy schedules given a latency (and often area) requirement (e.g., [16]–[18]). With single-V_{DD}, Fig. 4(a) already provides the optimal schedule and only requires two multipliers and one adder. As discussed above, multi-V_{DD} could implement either schedule in Fig. 4(b) or Fig. 4(c) based on energy vs. area trade-offs. PDVS implements the schedules in Fig. 4(b) or Fig. 4(c) with two and one adders, respectively (instead of the 3 and 2 adders required by multi-V_{DD}). To achieve the latter, the adder must switch quickly between the middle voltage level (V_{DDM}) and V_{DDH}, but the fine-grained temporal voltage scaling capabilities of PDVS make this possible. The additional energy required to make such voltage transitions does increase the total energy of this implementation, but as shown below, this overhead is small. As a result, PDVS can greatly improve the efficiency of a system within a particular energy/performance mode. Such solutions can be achieved with only slight modifications to existing multi-V_{DD} synthesis techniques to account for transition overheads (e.g., [19]), enabling PDVS to be easily incorporated into existing tool flows. The most closely related synthesis algorithm is based on a fine-grained header switch implementation similar to PDVS but does not use IC measurements for V_{DD}-switching overhead and does not compare results to static multi-V_{DD} [16].

Now we consider the efficiency of switching between such modes due to dynamic workload requirements. For example, consider the dynamic workload scenario in which a variable number of DFG iterations (reflected as the normalized workload axis in Fig. 6) must be performed in a given amount of time. Without any DVS capabilities, an architecture must operate at the maximum rate and enter a low-power sleep mode when it finishes early. As Fig. 6 shows, this provides only linear energy savings (i.e., a workload that is half of the maximum is executed with half of the maximum energy). The fine-grained *subblock* energy savings provided by Multi-V_{DD} and PDVS (due to component-level voltage assignment) appear in Fig. 6 as a downward shift of this curve, but the lack of DVS would still provide linear savings for reduced workloads.

As discussed above, DVS with either single- or multi-V_{DD} provides the ability to adjust the processing rate based on the workload but with some limitations. If the voltage can be selected from a continuous range and there is no energy or delay overhead to make the voltage transition (DVS ideal), quadratic energy savings can be achieved, with or without the subblock savings based on the number of V_{DD}s (dashed lines show the ideal DVS curves in Fig. 6). Unfortunately, existing dc–dc regulators that can provide

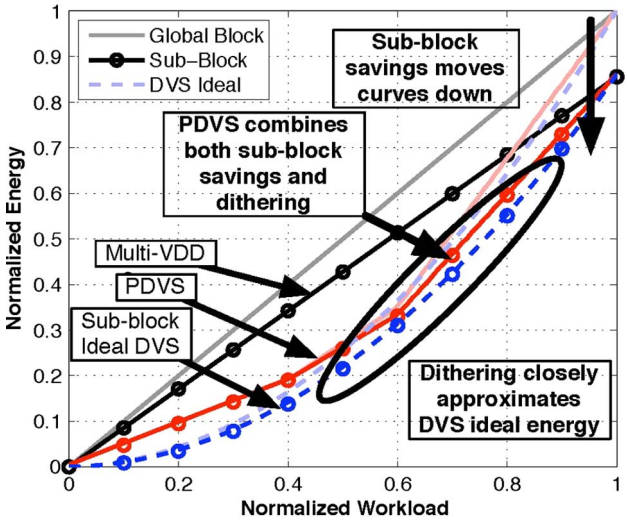


Fig. 6. ***PDVS achieves subblock energy savings and closely tracks ideal DVS with dithering.***

millivolt accuracy take hundreds of microseconds to adjust their voltage, making these ideal curves impractical except for taking advantage of very coarse grained changes in workload. Voltage dithering [26] is an approach that uses a small number of discrete voltages (multi-V_{DD}) to approximate the ideal curves. Dithering refers to the practice of operating for part of the time at a higher voltage and the remainder of the time at a lower voltage to average the performance at those endpoints to achieve any effective performance rate in between. Dithering thus leads to a linear characteristic that connects the dots between points on the ideal curve provided by the V_{DD} rails.

PDVS combines the benefits of fine-grained savings with a practical implementation of dithering to approach the energy-optimal processing rate. Instead of changing the voltages on large rails, the header switches provide fast and efficient voltage transitions for each component, enabling the system to effectively switch processing rates by implementing schedules with different latencies.

However, the flexibility of PDVS comes with an overhead. For example, PDVS can implement the schedule in Fig. 4(c) with only one adder, but there are finite energy and delay costs to switching between voltages, which may make this schedule energy-inefficient or un-implementable within the given latency constraints. The magnitudes of the voltage switching energy and delay overheads must be known in order to derive the conditions under which switching to a lower voltage is efficient. These conditions can then be used to inform architecture-level synthesis algorithms. We fabricated a test PDVS architecture (with a 32-bit Kogge–Stone adder and a 32-bit Baugh–Wooley multiplier—shown in Fig. 7) in 90 nm bulk CMOS to accurately measure these overheads.

Fig. 7 shows the relatively small area overhead (5% to 15%) introduced by PDVS header switches and level converter, in comparison to standard arithmetic component sizes, so the two primary sources of overhead that influence how algorithms are scheduled on the architecture are energy and delay. The overheads result from additional header switches, the level converters, and the charging of virtual V_{DD} nodes of components following a voltage switch. When a voltage switch occurs, the gates of header switches must be charged or discharged by control signals. The time to charge header gates results in a delay overhead, and the charge delivered to the switches and the virtual rail results in energy overhead. These overheads are opposing design trade-offs that are proportional to header width, which is determined by the priority of design metrics. A large header width results in lower channel resistance, decreasing the voltage drop caused by the header, but also increasing the delay and energy required to make the switch. Conversely, a small header size means that switch delay and energy is smaller, but the higher resistance of the channel increases the voltage drop across it, and consequently increases the delay of the ensuing combinational logic block [21].

The delay of transitioning to a higher voltage is critical, as the virtual V_{DD} rail must be charged up to the higher voltage. By properly sizing the header switches, the delay overhead can be reduced to less than one cycle [21]. Delay overheads factor into scheduling decisions by limiting voltage transition speed, and consequently, the amount of slack that can be saved within a schedule. As a result, an operation may not be able to run at the lowest possible processing rate, even though just enough slack time exists. However, this only occurs in schedules with very limited slack.

As discussed above, the point at which the energy overhead of switching is mitigated by processing at a lower energy rate is called the break-even time. An efficient transition occurs if the cost of switching to a lower energy rate, processing for a period of time, then switching back, is lower than remaining at the higher rate for the same amount of time. This concept is described by (3).

$$N \leq E_{ovh}/(E_H - E_L) \tag{3}$$

where N is the number of clock cycles, E_{ovh} is the lumped, round-trip overhead energy, and E_H and E_L are the energy per cycle at the high and low voltage, respectively.

Table 1 shows a summary of measured values for the fabricated 3-rail PDVS architecture. The calculation for break-even time, based on these numbers and the break-even equation, has shown that the time required to mitigate the energy cost of switching between any two voltages in the test architecture is less than the latency of a single operation for both the adder and the multiplier. This means the energy of completing one execution at a low voltage plus the round-trip energy of switching down and switching back up is less than one execution at the highest rate. The low energy overhead therefore justifies frequent down-switching in PDVS to take advantage of fine-grained energy savings, dithering, and rapid changes in workload. The more often that a down-switch can be scheduled, the

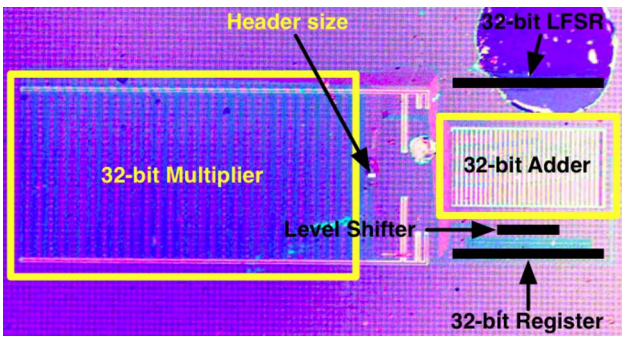


Fig. 7. *Die photo of the fabricated PDVS architecture. The relatively small area overhead of PDVS headers and level shifter is shown.*

Table 1 Summary of Measured Quantities From a 90 nm PDVS Test Chip

V_{DD} (V)	1	0.78	0.68
	Adder		
Delay (ns)	0.43	0.85	1.28
Active Energy per Op (pJ)	2.43	0.78	0.49
Total Energy per Op (pJ)	2.43	0.78	0.49
Nearest Delay (Adder cycles)	1	2	3
Switching Overhead (pJ)	-	1.12	1.61
Breakeven Cycles	-	0.55	0.71
	Multiplier		
Delay (ns)	2.96	4.31	6.5
Active Energy per Op (pJ)	61.28	35.59	28.38
Add'l Leakage per Op (pJ)	0.0013	0.0107	0.0055
Total Energy per Op (pJ)	61.29	35.6	28.38
Nearest Delay (Adder cycles)	7	11	16
Switching Overhead (pJ)	-	6.71	9.67
Breakeven Cycles	-	1.22	1.39

lower the average voltage that is used, and consequently, the lower the total execution energy.

One further consideration in PDVS is noise created on the global V_{DD} rails that results from switching the headers for the blocks. Existing work has characterized this type of rail bounce in power gated systems [20]. While it is a problem that requires attention, techniques are available to reduce rail bounce (e.g., [20]). Furthermore, the PDVS scheme uses smaller switches (for local blocks, not entire cores) and switches internal voltages by a smaller voltage difference (e.g., V_{DDL} to V_{DDH}, not $\sim V_{DD}$ to ground), leading to lower currents and less charge transfer from the global rails. This reduces the problem relative to conventional power gating and should make existing solutions adequate for preventing unreasonable rail bounce.

So far, we have focused on how PDVS saves energy for relatively high performance scenarios. UDVS [9] also supports subthreshold operation by either adjusting the dc–dc converter to a lower voltage or by using an additional header in parallel with the other PDVS headers to hop to a subthreshold supply voltage (note that this voltage value could double as an appropriate standby mode voltage for higher performance blocks that need to retain data in standby mode). In either case, the dramatic difference in processing rate between subthreshold and strong inversion means that frequent switching (e.g., within one DFG) between these modes is not very useful. Instead, the transition to and from subthreshold more likely would occur as an infrequent mode change.

The large difference in voltage from V_{DDH} to the subthreshold voltage makes the efficiency of the header switches and other transistors in the active block an important issue. Specifically, one must carefully consider how to connect the bulk terminal of the transistors in the active circuit block and of the header switches. Fig. 8 shows

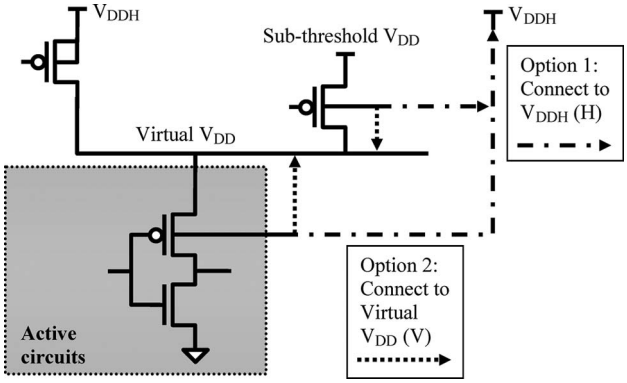


Fig. 8. *Options for bulk connection of active circuits and subthreshold header switch. They can either connect to V_{DDH} (denoted H) or to the virtual V_{DD} rail (denoted V).*

the options as either the highest voltage in the circuit, V_{DDH} (denoted H), or the virtual V_{DD} rail (denoted V). The N-wells in the active circuit can tie to either of these locations, and the bulk terminal of the header whose source connects to the subthreshold V_{DD} can also tie to either place. To avoid turning on the diode between the drain/source and the bulk, the drain/source terminal can never be more than a few hundred millivolts above the bulk voltage. Tying all of the PMOS bulk terminals to V_{DDH} prevents this condition, but it severely reverse biases the header and the active circuit when the header connecting the active circuit to the subthreshold V_{DD} is on.

Fig. 9 shows the delay and energy of a 32b adder connected to headers as in the schematic from Fig. 8. In Fig. 9(a), the left two bars show the delay of the addition at 0.3 V when the bulk terminals of the PMOS transistors in the adder connect to V_{DDH}. The reverse body bias

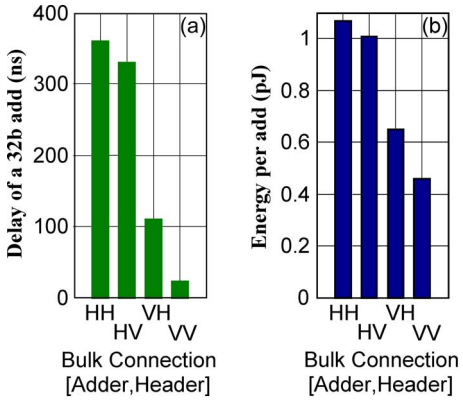


Fig. 9. *Delay (a) and energy (b) of a 32b adder in 90 nm CMOS at $V_{DD} = 0.3$ V. The bulk of the adder and header (Adder, Header) can either connect to V_{DDH} (denoted H) or to the virtual V_{DD} rail. Thus, HV indicates the adder bulk tied to V_{DDH} and the header bulk tied to virtual V_{DD}.*

increases the PMOS V_T and slows the operation substantially compared to the right two bars (VH and VV), for which the adder PMOS bulks connect to the virtual supply rail. Since that rail equals 0.3 V in this case, the adder sees no body bias and operates with much less delay. The effect of the header's body connection on the delay primarily results from substantial rail bounce. When the header bulk connects to V_{DDH}, the virtual rail takes a long time to recover to the full V_{DD} due to the increased resistance of the header. This droop in the effective supply voltage slows down back-to-back operations. Connecting the bulk to the virtual rail reduces the droop on the supply. Fig. 9(b) shows that the energy also dramatically reduces when both bulks connect to the virtual rail. Despite the lower absolute value of PMOS V_T, the shorter delay reduces leakage energy and thus lowers overall energy. This adaptive body connection does not require additional control signals.

This analysis shows that the bulk terminal of the subthreshold header should connect to its drain at the virtual V_{DD} rail. The additional capacitive load on the virtual rail will be a small fraction of its total capacitance for reasonable header sizes. In a PDVS system that supports UDVS, the bulk terminals of the active circuit (e.g., the adder in our example) should also connect to the virtual rail. The combined capacitance of those N-wells will increase the virtual rail load by roughly 10% to 15%, so this decision will impact the speed and energy overheads of rapid switching between higher supply voltages in PDVS.

It is also interesting to consider PDVS in the context of the increasing process variations resulting from technology scaling. Given that the header sizes are quite large, they will be relatively unaffected by variations, so the impact of variations will be no worse in PDVS than in other architectures. In fact, PDVS can be used to help address process variations by selecting voltages that are appropriate for components based on each component's characteristics, similar to what has been proposed in [22].

Standard processor DVS has been shown to improve energy efficiency in a number of systems, and PDVS promises to increase the applicability of DVS to achieve more frequent and larger energy savings. For example, body area sensor networks (BASNs) used in biomedical and healthcare applications can make use of DVS to save energy as operations change with time [23]. Specifically, accelerometer-based BASNs are used to monitor tremor in Parkinson's disease patients. On-node signal processing is performed to reduce the amount of data that must be wirelessly transmitted [24], but these processing needs vary dynamically due to the transient nature of tremor. When tremor is present, the tremor assessment (based on Teager Energy [25]) requires that the on-node processor operate in a high-throughput mode, but the processor can otherwise operate in an ultralow-power mode, using simple thresholding to determine if a tremor might be present. Efficient transitions between these modes is necessary to reap the potential energy benefits of DVS while still meeting the application's requirements, making PDVS an ideal solution for this type of application.

This section addressed the issue of flexibility to respond when the performance requirements vary for a specific operation or set of operations. A different sort of flexibility becomes necessary when a system needs to implement different functions at different times. The next section deals with hardware flexibility.

IV. A SUBTHRESHOLD FPGA FOR HARDWARE FLEXIBILITY

In addition to requiring operation at varying speeds, many ULP applications would benefit from supporting varying types of operations. For example, wireless microsensors sense data, process it using signal processing algorithms, and then communicate the data wirelessly across a network. It is easy to conceive that the signal processing algorithms on the chip would vary based on the type of data coming from the sensor, the sensor's environment, or the sensor's specific application. Building flexibility into the hardware to support these changes comes at the cost of energy inefficiency. In this section, we examine this trade-off in the context of subthreshold, ULP applications and motivate the need for subthreshold reconfigurable logic such as field programmable gate arrays (FPGAs).

This well known trade-off between flexibility and efficiency appears prominently in a comparison of conventional hardware paradigms [29], [30]. Fig. 10 shows how different types of hardware compare in flexibility and efficiency [30]. The most efficient hardware essentially is hardwired to do its specific task or tasks (e.g., application-specific IC, or ASIC). ASICs achieve very efficient operation, but they can only perform the function for which they were originally defined. Examples of hardwired implementations in subthreshold circuits include [4], [5], [44], [45]. The most flexible category of hardware is general purpose processors (GPPs). GPPs exhibit poor energy efficiency due to the overhead of fetching and decoding the

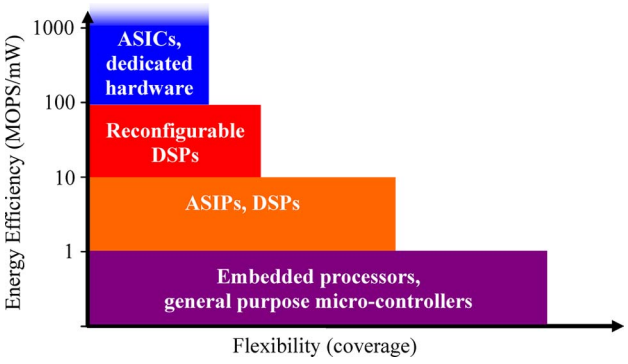


Fig. 10. *Trade-off of energy efficiency with functional flexibility in strong inversion (adapted from [30]).*

instructions that define the actual operation. Moreover, the operations executed by many GPPs are fairly light weight, requiring many instructions to complete a larger operation, like an FFT or filter. Several subthreshold GPPs provide energy per instruction nearing 1 pJ per operation, but they also tend to use small instruction sets [37]–[40].

Examples of subthreshold systems reveal a similar trend as their above-threshold counterparts. For example, the subthreshold microcontroller in [37] consumes as low as 2.6 pJ/instruction. The ASIC implementation of a JPEG coprocessor in [44] consumes 1.3 pJ/frame for VGA JPEG encoding. The numbers for energy/operation are similar, but the GPP operations on the microcontroller (e.g., instructions), tend to be simple integer computations like addition. In comparison, a complete JPEG encoding would take many (100 s or 1000 s) instructions on one of these simple GPPs to complete. Of course, the GPP can perform a much broader range of tasks than the JPEG encoder, so this comparison exemplifies the trade-off between energy efficiency and flexibility.

Existing subthreshold architectures that target ULP applications leave a noticeable gap in the flexibility/efficiency space. For above threshold, reconfigurable logic like FPGAs fill this space. We propose that subthreshold implementations of reconfigurable logic would improve the scope of ULP systems for two reasons. First, the gap between energy efficiency of ULP ASICs and GPPs is quite large. As a result, subthreshold GPPs may consume too much power while implementing computationally intensive algorithms in some ULP applications. ASICs are much more efficient, but they limit the extent to which an application can be reconfigured efficiently. Due to the strict energy budgets of ULP applications, the best existing solution for many of them at this time appears to be subthreshold ASICs.

The second motivation for subthreshold FPGAs arises from the cost of ASIC development. Process scaling has increased the engineering costs and the fabrication costs of ASICs dramatically. The difficulties of designing to deal with variations in subthreshold can easily require multiple fabrication runs to produce working parts with acceptable yield. Furthermore, while some ULP applications like RFIDs are high volume, many others like sensing applications are relatively low volume, and their stringent energy constraints and specific processing needs make GPPs an inappropriate solution. On the other hand, targeting an ASIC or application-specific instruction set processor (ASIP) to low volume applications requires the nonrecoverable engineering costs to design and fabricate a new chip for each new application, making the cost prohibitive.

Reconfigurable logic fills the gap between GPPs and ASICs in the conventional hardware trade-off space, so it provides a natural candidate to improve the energy efficiency of subthreshold circuits while retaining a high degree of flexibility. One common form of reconfigurable logic is the field-programmable gate array (FPGA). A conventional FPGA comprises an array of configurable logic blocks (CLBs), which can implement a variety of functions, connected by circuit-switched programmable interconnect. Fig. 11 shows this basic structure of an FPGA. To reconfigure the FPGA, the user streams a set of configuration bits onto the device. These configuration bits set the function of the CLBs by filling look up tables (LUTs) that map inputs to desired outputs. The configuration bits also determine the routing of signals between CLBs by turning on and off appropriate switches in the interconnect fabric. After programming, the FPGA acts essentially as an ASIC in the sense that it provides a hardware instantiation of the desired function. The resulting hardwired circuit avoids the overhead (e.g., of fetching instructions, etc.) inherent to processors and provides improved energy efficiency.

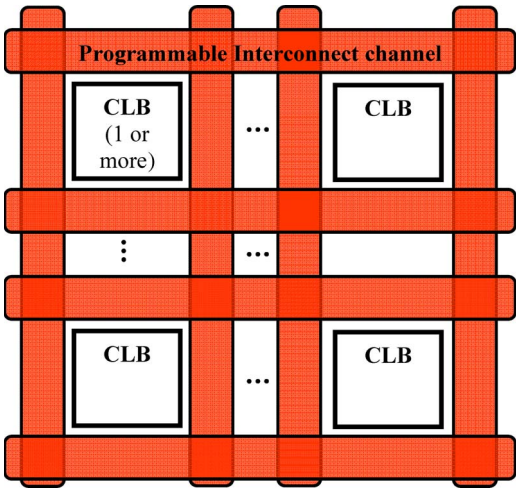


Fig. 11. *Basic architecture of an FPGA consisting of configurable logic blocks (CLBs) connected by a programmable interconnect fabric.*

The flexibility and energy efficiency of FPGAs make them appealing for subthreshold operation. Their appeal increases considering the high cost of custom circuit design in modern processes coupled with the relatively low volume of chips required for many of the ULP applications. Designing one custom FPGA for subthreshold use would potentially provide an energy efficient platform that could retarget new applications with ULP constraints. The device's flexibility would reduce time to market for emerging ULP products and would serve enough application needs to achieve high volume and low-cost production. In the next section, we examine the circuit level challenges to implementing a subthreshold compatible FPGA for ULP applications.

A. Subthreshold Reprogrammable Logic

In this section, we explore the challenges facing subthreshold FPGA design and present simulation results from an initial implementation of a subthreshold FPGA. As

previously described, FPGAs provide the flexibility, short time-to-deployment, and low unit cost that are ideal for many ULP applications. However, conventional FPGAs are notoriously high power. Commercial FPGAs in 90 nm technology can consume on the order of 1–10 W, and leakage power can be a large fraction of the total [31]. The major reason for this large power consumption is the focus of FPGA vendors on reducing delay and area to compete with high-performance ASICs. The emergence of many commercial low-power applications has produced some effort to reduce FPGA power. For example, applying architectural and circuit-level power saving methods can reduce power by 1 to 2 orders of magnitude relative to commercial FPGAs [31]. The resulting devices still consume power in the tens of milliWatts, which exceeds constraints for ULP applications. By modifying existing low-power FPGA topologies to operate in the subthreshold region, we can decrease power consumption to ULP levels.

B. Subthreshold FPGA Design and Challenges

Subthreshold FPGA design faces a combination of subthreshold circuit challenges and problems inherent to FPGA architectures. First, subthreshold circuit design inherently has many challenges including low I_{on}/I_{off} ratio, exponential impact of V_T variation on current, reduced SRAM noise margin, etc. [4], [34]–[36], as we described in Section II. All of these challenges appear in a subthreshold FPGA design as well, although they manifest themselves differently due to the unique FPGA architecture. To clearly understand these challenges, we have designed and simulated an FPGA that can operate into the subthreshold region in a commercial 90 nm bulk CMOS technology. Again, the goal of an ULP FPGA design is to minimize energy consumption in the circuits.

The basic architecture of our subthreshold FPGA matches the conventional topology in Fig. 12. Tools for programming an FPGA (e.g., mapping a function onto the underlying hardware) are essential for making FPGAs usable, so keeping with the conventional topology allows us to leverage existing toolsets (e.g., [28]). The CLBs each have four basic logic elements (BLEs). Each BLE consists of four 4-input LUTs, a register, and programmable multiplexors to control the data flow. The interconnect fabric in between the CLBs allows for connections into and out of the CLB using *connection boxes*. Programmable *switch boxes* provide the connectivity for signals that travel from one CLB to another. Each of the switch boxes acts as a full crossbar to support connectivity in all directions. The routing channels between CLBs in Fig. 12 have a width of 10 for clarity in the illustration, but most FPGAs support much wider channel widths of 60 to 100.

Three major challenges stand out for the subthreshold FPGA design. These are variation, interconnect, and memory. Variations threaten to disrupt any subthreshold design, but the FPGA encounters variation effects in a few key ways. As with SRAM, the FPGA contains a large

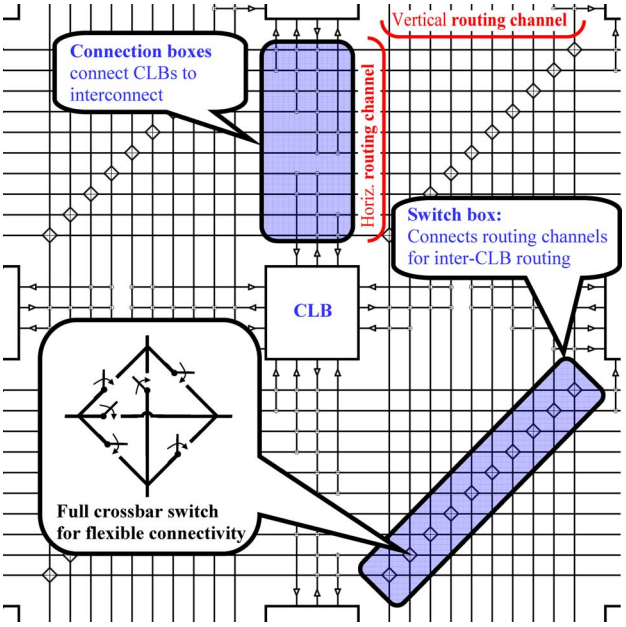


Fig. 12. *Basic architecture of an FPGA consisting of configurable logic blocks (CLBs) connected by a programmable interconnect fabric.*

number of parallel elements that ideally behave in an identical fashion. In reality, process variations cause these circuit components to vary, and the spread of this variation in circuit level parameters like delay and I_{on}/I_{off} expands in the subthreshold region. This wide variation makes careful statistical simulation important to ensure functioning circuits. The inherent programmability of the FPGA fabric provides one tool to address this problem. The process of mapping a function onto the FPGA can include a step to avoid problematic sections of the FPGA (e.g., a weak driver or unstable register) by assigning functionality to different regions of the chip. The FPGA tool chain would have to change to take advantage of the built-in redundancy of FPGAs in this way. Specifically, the place and route step would need to account for variation data measured from the specific targeted part so that the synthesized design could avoid placing critical paths on gates that are slowed excessively by variation. This sort of approach would require test data from each FPGA die to be available during the synthesis process. Variation also creates a major challenge in the FPGA's interconnect structures.

The second major challenge for subthreshold FPGAs involves designing a functional and practical interconnect network. The interconnect network is the dominant feature of traditional FPGAs as well. Although the numbers vary from design to design, FPGAs typically dissipate 60%–70% of their power in the interconnect network (e.g., wires, connection boxes, routing switches and buffers), 10%–20% in the clock network, and 5%–20% in logic (e.g., [31]–[33]). This breakdown indicates that a focus on clocking and interconnect is necessary. These are

two areas on which very little previous subthreshold work has focused.

First, the clock network for an FPGA extends across the entire FPGA fabric and drives all of the registers in the design. This large, distributed network can consume a significant amount of power. Furthermore, driving the large capacitive load of the clock network with buffers operating in subthreshold presents a significant problem. Variation in the buffers can lead to substantial differences in drive strength that are essentially amplified by the large loads, leading to potentially large clock skew across the FPGA fabric. This problem is not substantially different for FPGAs relative to ASICs, but clock distribution has not received much attention for subthreshold designs. One reason for this may be that most subthreshold processors to date have been fairly small, allowing for monolithic clock buffers to drive the whole clock grid within generating too much skew. Since FPGA fabrics may grow to cover much larger total areas, a power efficient method to reduce skew is important for viable subthreshold FPGA designs.

The same issues that make clock power and variability problematic also plague the interconnect network. In subthreshold, the transistor drive current decreases while the wire-dominated capacitive load of the interconnect stays largely the same. The wire and parasitic capacitance on a single interconnect segment (and an interconnect net usually consists of many segments) roughly equals 25×–50× the load of a fan-out of four inverter. Simply upsizing the driver does not offer a practical solution since the interconnect fabric already consumes roughly 75% of FPGA area. Variability in the drivers also increases the variation of this large interconnect delay. To make matters worse, the leakage onto the interconnect paths from *off* devices in the switch boxes and connection boxes fight against the on-current of the drivers, which is weakened in subthreshold. Since the main goal in a subthreshold design is to reduce energy consumption, decreasing the numbers of buffers in the interconnect is a possible approach to lower the total switched capacitance. Strong inversion FPGAs could use transmission gates or single-transistor passgates in the interconnect to reduce switched capacitance and leaking area, but these structures in subthreshold combine with the lower I_{on}/I_{off} ratios and result in reduced output swing on the interconnect lines. The lower swing can cause static current in the receiving circuits.

Fig. 13 shows a simulation of an interconnect link in subthreshold ($V_{DD} = 0.4$ V) that illustrates some of these problems. The interconnect path in this simulation spans ten wire segments and ten switch boxes, which use transmission gates to complete the crossbar connections. From an energy point of view, this structure would be nice to use since it decreases the switching and leakage energy that would be consumed by using more buffers. In strong inversion, passgates often replace transmission gates to reduce transistor counts, but series-connected passgates lead to very poor output swing and speed in subthreshold. Fig. 14(a) shows an interconnect link with some switchboxes that use tristate buffers as repeaters, and Fig. 14(b) shows the composition of those switchboxes. Fig. 15 shows simulation results for the same ten segment interconnect line with buffers every fourth link (a), every other link (b), and every switchbox (c). Placing buffers in each switchbox decreases the variability and increases the speed in the 10-segment link, as Table 2 shows. The cost of this approach is additional area in the switchboxes, but it may

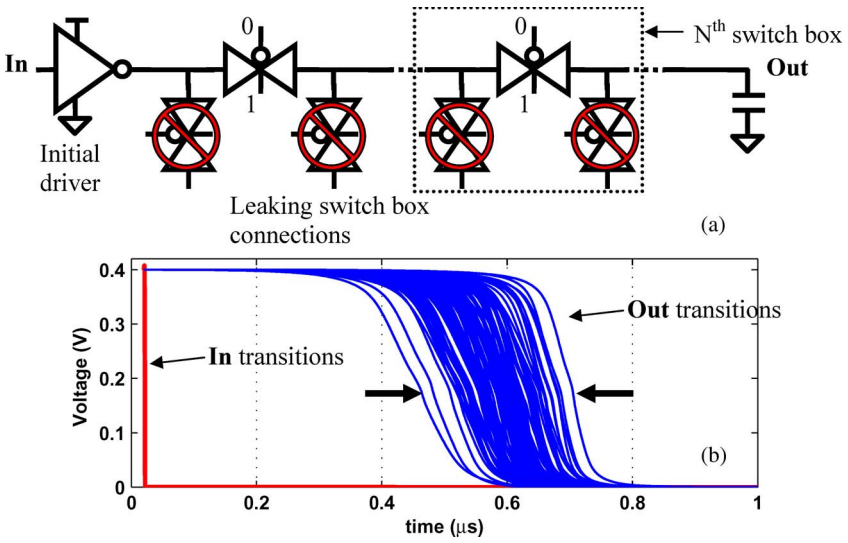


Fig. 13. ***Interconnect link across ten segments with transmission gates in the switch boxes (a). Monte Carlo simulation at 0.4 V shows large delay variation (b).***

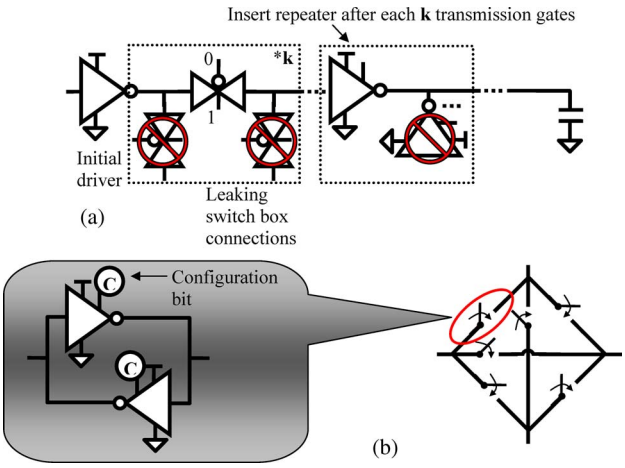


Fig. 14. *Interconnect link of ten segments with tristate inverter acting as a repeater after k switch boxes that use transmission gates (a). Diagram of switch box crossbar using repeaters (b).*

be essential in subthreshold to combat the significant effect of variation.

Even after deploying repeaters in every switchbox, the delay and energy of the interconnect network continue to dominate the overall FPGA metrics in subthreshold. Fig. 16 shows the breakdown of delay and energy for our simulated FPGA fabric running a representative bench-

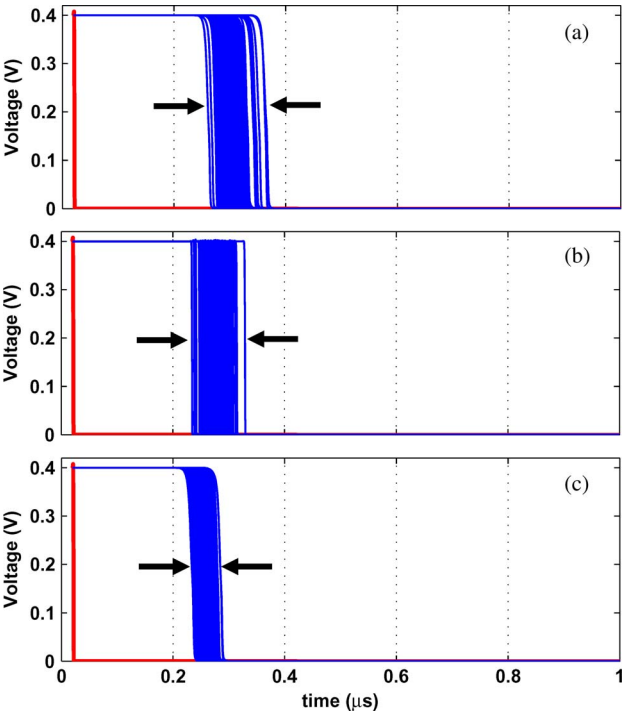


Fig. 15. *Monte-Carlo simulation at 0.4 V for a ten segment link with repeater-based switch boxes every 4th segment (a), every other segment (b), and every segment (c).*

Table 2 Delay Statistics for Buffered Interconnect Links

Buffers	μ_{DELAY} (ns)	σ_{DELAY} (ns)	σ/μ
None	560	45	8.0%
Every 4th	280	21	7.5%
Every other	260	18	6.9%
Every	240	12	5.0%

mark configuration [28]. Clearly, the design of reliable, low power, and fast interconnect poses a challenging area of research for subthreshold programmable logic.

The large amount of memory on an FGPA poses the third major challenge area for subthreshold reconfigurable logic. The memory on an FPGA appears in several different contexts. First, the CLBs contain registers that load the clock net and that lie on the critical path of the programmed functions. The important metrics for these registers do not substantially differ from those of registers in other subthreshold circuit contexts. Second, the LUTs in the FPGA CLBs consist of SRAM bit cells. These cells hold the data values that map inputs to arbitrary functions to provide the heart of the FPGA's programmability. These bitcells usually only require a write access during reprogramming of the FPGA, but they essentially provide continuous read access during FPGA operation. Furthermore, the inputs select one bit to read from the LUT, so this read access lies directly on the critical path through the CLB. The LUT memory thus requires a design that focuses on high-speed, low-power reads. Finally, a large volume of SRAM bit cells distributed across the FPGA fabric provide the configuration bits that store the programmed function of the device. These bits drive multiplexors and switch boxes to configure the hardware for the proper function. A direct connection from the storage node of one of these bits to the gate terminals of other transistors usually provides the required "read access," and write accesses occur only during reprogramming. This means that the major metrics for configuration bits are data retention and low leakage. Since the configuration bits do not switch after programming, they can use high V_T transistors to reduce their leakage. For an FPGA design that will operate exclusively in subthreshold and that has a large amount of leaking circuits

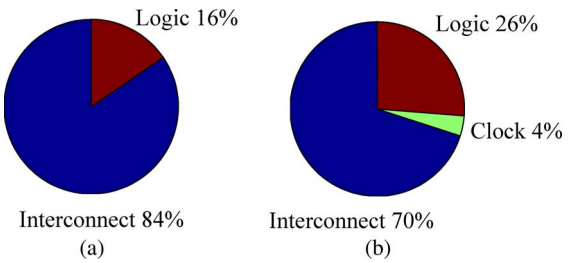


Fig. 16. *The breakdown of delay (a) and energy (b) in simulations of an FPGA at a subthreshold voltage of 0.4 V for a representative function.*

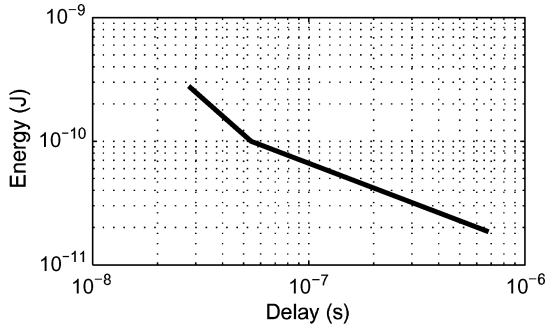


Fig. 17. ***Simulated energy vs. delay for a function mapped onto a subthreshold FPGA fabric in 90 nm CMOS as V_{DD} scales into subthreshold.***

and/or long sleep times, choosing an older technology makes the most sense from an energy (and battery lifetime) point of view because it reduces the dominant leakage component of the circuits (e.g., [46]). Of course, these energy savings would need to be weighed against economic considerations of using larger feature sizes.

Fig. 17 shows the energy and delay of a complete FPGA fabric running a representative function at three supply voltages ranging from the nominal voltage to the subthreshold value of 0.4 V. Overcoming the challenges that face a subthreshold FPGA design clearly provide the anticipated energy savings at low voltage. The subthreshold FPGA offers a flexible platform for ULP applications that gives low energy per operation and cheap, rapid redeployment for alternative applications.

V. CONCLUSION

When the functional or speed requirements of an application vary, building flexibility into the system saves power. To address varying speed requirements, this paper describes a circuit/architecture codesign approach called panoptic DVS (PDVS) that uses multiple voltage supplies and power switches to apply the optimal V_{DD} for the required performance at the block level. Measurements of a test chip confirm that the overhead of this approach remains low, allowing for efficient energy-performance scaling, including break even times for mode transitions of less than a single operation. To address varying functional modes either within one application or across applications, we propose that a subthreshold FPGA provides a good trade-off of flexibility and energy efficiency. The reconfigurable hardware saves energy relative to general purpose subthreshold microcontrollers, and the flexibility of the FPGA makes it suitable for rapid redeployment in new applications with low unit cost. ■

Acknowledgment

The authors acknowledge Jiajing Wang, Kyle Ringgenberg, and Liang Di for their work supporting the results described in this paper.

REFERENCES

[1] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 114–117, Apr. 1965.

[2] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein, "Scaling, power, and the future of CMOS," in *Proc. IEEE Int. Electron Devices Meeting*, Dec. 2005, pp. 9–15.

[3] A. Wang, A. Chandrakasan, and S. Kosonocky, "Optimal supply and threshold scaling for sub-threshold CMOS circuits," in *IEEE Comput. Soc. Annu. Symp. VLSI*, Apr. 2002, pp. 7–11.

[4] B. H. Calhoun, A. Wang, and A. Chandrakasan, "Modeling and sizing for minimum energy operation in sub-threshold circuits," *IEEE J. Solid-State Circuits*, vol. 40, no. 9, pp. 1778–1786, Sep. 2005.

[5] A. Wang and A. Chandrakasan, "A 180 mV FFT processor using subthreshold circuit techiques," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2005, pp. 292–293.

[6] B. H. Calhoun and A. Chandrakasan, "A 256 kb sub-threshold SRAM in 65 nm CMOS," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2006, pp. 628–629.

[7] S. Khanna and B. H. Calhoun, "Serial sub-threshold circuits for ultra-low-power systems," in *Proc. Int. Symp. Low Power Electron. Design*, Aug. 2009.

[8] B. H. Calhoun, D. D. Daly, N. Verma, D. Finchelstein, D. D. Wentzloff, A. Wang, S.-H. Cho, and A. Chandrakasan, "Design considerations for ultra-low energy wireless microsensor nodes," *IEEE Trans. Computers*, vol. 54, no. 6, pp. 727–740, Jun. 2005.

[9] B. H. Calhoun and A. Chandrakasan, "Ultra-Dynamic Voltage Scaling (UDVS) using sub-threshold operation and local voltage dithering," *IEEE J. Solid-State Circuits*, vol. 41, no. 1, pp. 238–245, Jan. 2006.

[10] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," *JSSCC*, vol. 35, no. 11, pp. 1571–1580, 2000.

[11] S. Lee and T. Sakurai, "Run-time voltage hopping for low-power real-time systems," in *Proc. Design Automation Conf.*, 2000, pp. 806–809.

[12] D. E. Lackey, P. S. Zuchowski, T. R. Bednar, D. W. Stout, S. W. Gould, and J. M. Cohn, "Managing power and performance for system-on-chip designs using voltage islands," in *Proc. Int. Conf. Comput. Aided Design*, 2002, pp. 195–202.

[13] J. Hu, Y. Shin, N. Dhanwada, and R. Marculescu, "Architecting voltage islands in core-based system-on-a-chip designs," in *Proc. Int. Symp. Low Power Electron. Design*, 2004, pp. 180–185.

[14] V. Gutnik and A. Chandrakasan, "An efficient controller for variable supply-voltage low power processing," in *Proc. Symp. VLSI*, 1996, pp. 158–159.

[15] G. Wei, O. Trescases, and W. T. Ng, "A dynamic voltage scaling controller for maximum energy saving across full range of load conditions," in *Proc. Conf. Electron Devices Solid-State Circuits*, 2005, pp. 371–374.

[16] D. Chen, J. Cong, and J. Xu, "Optimal simultaneous module and multivoltage assignment for low power," *ACM Trans. Design Autom. Electron. Syst.*, vol. 11, no. 2, pp. 362–386, Apr. 2006.

[17] R. Puri, L. Stok, J. Cohn, D. Kung, D. Z. Pan, D. Sylvester, A. Srivastava, and S. H. Kulkarni, "Pushing ASIC performance in a power envelope," in *Proc. Design Autom. Conf. (DAC)*, 2003, pp. 788–793.

[18] A. Andrei, P. Eles, Z. Peng, M. T. Schmitz, and B. M. Al Hashimi, "Energy optimization of multiprocessor systems on chip by voltage selection," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 3, pp. 262–275, Mar. 2007.

[19] M. Putic, L. Di, B. H. Calhoun, and J. Lach, "Panoptic DVS: A fine-grained dynamic voltage scaling framework for energy scalable CMOS design," in *Proc. Int. Conf. Comput. Design*, 2009.

[20] S. Kim, S. V. Kosonocky, and D. R. Knebel, "Understanding and minimizing ground bounce during mode transition of power gating structures," in *Proc. Int. Symp. Low Power Electron. Design*, 2003, pp. 22–25.

[21] L. Di, M. Putic, B. Calhoun, and J. Lach, "Power switch characterization for fine-grained dynamic voltage scaling," in *Proc. Int. Conf. Comput. Design*, 2008, pp. 605–611.

[22] X. Liang, G.-Y. Wei, and D. Brooks, "ReVIVaL: A variation-tolerant architecture using voltage interpolation and variable

latency," in *Proc. Int. Symp. Comput. Archit.*, 2008, pp. 191–202.

[23] H. C. Powell, Jr., A. T. Barth, and J. Lach, "Dynamic voltage-frequency scaling in body area sensor networks using COTS components," in *Proc. Int. Conf. Body Area Networks*, 2009.

[24] M. A. Hanson, H. C. Powell, Jr., R. C. Frysinger, D. S. Huss, W. J. Elias, and J. Lach, "Teager energy assessment of tremor severity in clinical application of wearable inertial sensors," in *Proc. IEEE-NIH Life Sci. Syst. Appl. Workshop*, 2007, pp. 191–194.

[25] J. F. Kaiser, "On a simple algorithm to calculate the energy of a signal," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, 1990, vol. 1, no. 3–6, pp. 381–384.

[26] V. Gutnik and A. Chandrakasan, "Embedded power supply for low-power DSP," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 5, no. 4, pp. 425–435, Dec. 1997.

[27] R. Keyes, "The effect of randomness in the distribution of impurity atoms on FET threshold," *Appl. Phys. A: Mater. Sci. Process.*, vol. 8, pp. 251–259, 1975.

[28] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," in *Proc. Int. Conf. Field Programmable Logic Appl.*, 1997, pp. 213–222.

[29] J. M. Rabaey, A. Abnous, Y. Ichikawa, K. Seno, and M. Wan, "Heterogeneous reconfigurable systems," in *Proc. Workshop Signal Process. Syst.*, 1997, pp. 24–34.

[30] H. Zhang, V. Prabhu, V. George, M. Wan, M. Benes, A. Abnous, and J. M. Rabaey, "A 1-V heterogeneous reconfigurable DSP IC for wireless baseband digital signal processing," *IEEE J. Solid-State Circuits*, vol. 35, no. 11, pp. 1697–1704, Nov. 2000.

[31] V. George, H. Zhang, and J. Rabaey, "The design of a low energy FPGA," in *Proc. Int. Symp. Low Power Electron. Design*, 1999, pp. 188–193.

[32] L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic power consumption in the Virtex-II FPGA family," in *Proc. Int. Symp. Field-Programmable Gate Arrays*, 2002, pp. 157–164.

[33] F. Li, D. Chen, L. He, and J. Cong, "Architecture evaluation for power-efficient FPGAs," in *Proc. Int. Symp. Field-Programmable Gate Arrays*, 2003, pp. 175–184.

[34] A. Wang, B. H. Calhoun, and A. Chandrakasan, *Sub-Threshold Design for Ultra-Low-Power Systems*. New York: Springer, 2006.

[35] J. F. Ryan, J. Wang, and B. H. Calhoun, "Analyzing and modeling process balance for sub-threshold circuit design," in *GLSVLSI*, Mar. 2007, pp. 275–280.

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

[37] 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 *Proc. Symp. VLSI Circuits*, 2006, pp. 154–155.

[38] 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 *Proc. Symp. VLSI Circuits*, 2008, pp. 188–189.

[39] J. Kwong, Y. Ramadass, N. Verma, and A. Chandrakasan, "A 65 nm sub-V_t microcontroller with integrated SRAM and switched capacitor DC–DC converter," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 115–126, Jan. 2009.

[40] S. Jocke, J. Bolus, S. N. Wooters, A. D. Jurik, A. C. Weaver, T. N. Blalock, and B. H. Calhoun, "A 2.6-μW sub-threshold mixed-signal ECG SoC," in *Proc. Symp. VLSI Circuits*, 2009.

[41] N. Verma and A. Chandrakasan, "A 256 kb 65 nm 8 T subthreshold SRAM employing sense-amplifier redundancy," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 141–149, Jan. 2008.

[42] T. H. Kim, J. Liu, J. Keane, and C. H. Kim, "A 0.2 V, 480 kb subthreshold SRAM with 1 k cells per bitline for ultra-low-voltage computing," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 518–529, Feb. 2008.

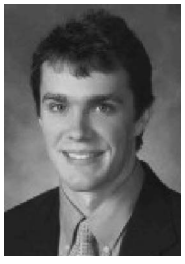
[43] J. P. Kulkarni, K. Kim, and K. Roy, "A 160 mV robust Schmitt trigger based subthreshold SRAM," *IEEE J. Solid-State Circuits*, vol. 42, no. 10, pp. 2303–2313, Oct. 2007.

[44] Y. Pu, J. P. de Gyvez, H. Corporaal, and Y. Ha, "An ultra-low-energy/frame multi-standard JPEG co-processor in 65 nm CMOS with sub/near-threshold power supply," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2009, pp. 146–147.

[45] C. Kim, H. Soeleman, and K. Roy, "Ultra-low-power DLMS adaptive filter for hearing aid applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 6, Dec. 2003.

[46] B. H. Calhoun, J. Bolus, S. Khanna, A. D. Jurik, A. F. Weaver, and T. N. Blalock, "Sub-threshold operation and cross-hierarchy design for ultra low power wearable sensors," in *Proc. Int. Symp. Circuits Syst.*, 2009.

ABOUT THE AUTHORS

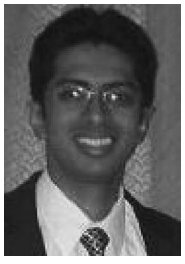


Benton Highsmith Calhoun (Member, IEEE) received the B.S. degree in electrical engineering from the University of Virginia, Charlottesville, in 2000 and the M.S. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 2002 and 2006, respectively.

In January 2006, he joined the faculty at the University of Virginia as an Assistant Professor in the Electrical and Computer Engineering Department. His research interests include low-power digital circuit design, subthreshold digital circuits, SRAM design for end-of-the-roadmap silicon, variation tolerant circuit design methodologies, low-power configurable logic, and low-energy electronics for medical applications. Dr. Calhoun has been the PI or co-PI for more than 17 grants, including a DARPA Young Faculty Award. He has published over 40 refereed papers, including one Best Paper Award, and is a coauthor of *Sub-Threshold Design for Ultra Low-Power Systems* (Springer, 2006).

Joseph F. Ryan received B.S. degrees in computer science and engineering and electrical engineering from Ohio State University, Columbus, in 2006. He is working toward the Ph.D. degree in electrical engineering as part of the Robust Low Power VLSI Circuits group at the University of Virginia, Charlottesville.

His main research interests are process variation tolerant circuit design, subthreshold circuit design, and reconfigurable computing.



Sudhanshu Khanna received the B.Eng. degree in electronics and communication from Delhi University in 2005. He is currently working towards the Ph.D. degree in electrical engineering at the University of Virginia, Charlottesville.

From 2005 to 2008, he was with Texas Instruments working on low-power circuit design for portable applications. His research interests are in subthreshold circuit design, low-power memory design, and variation tolerant design methodologies.

Mateja Putic (Member, IEEE) received the B.Sc. degree in computer engineering and the M.Sc. degree in electrical engineering from the University of Virginia, Charlottesville.

As part of his graduate thesis, he developed a hardware/software codesign methodology and demonstrative chip in a 90 nm process for ultralow-power embedded systems, achieving energy reduction as high as 37% over traditional designs. In addition, he has produced three academic publications related to low-power embedded design. More recently, he has acted as the Chief Product Designer for Energy Guardian, a smart home energy monitoring venture based in Charlottesville. He currently holds the position of Research Engineer in the Intelligent Systems Group at Luna Innovations, where he is developing an innovative embedded wireless condition-based sensing platform.

Mr. Putic is a member of Eta Kappa Nu.

John Lach (Senior Member, IEEE) received the B.S. degree from Stanford University, Stanford, CA, in 1996 and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Los Angeles, in 1998 and 2000, respectively.

Since 2000, he has been on the faculty of the Charles L. Brown Department of Electrical and Computer Engineering at the University of Virginia, Charlottesville, where he has held the rank of Associate Professor since 2006. His research interests are body sensor networks and other wearable technologies for biomedical and healthcare applications, integrated-circuit design techniques, dynamically adaptable and real-time embedded systems, fault and defect tolerance, safety-critical system design and analysis, general-purpose and application-specific processor design, and field-programmable gate arrays (FPGAs).

Dr. Lach has been the PI or co-PI for more than 20 grants and has published over 70 refereed papers, including two Best Paper Awards. He is an Associate Editor for the IEEE TRANSACTIONS ON COMPUTERS and IEEE TRANSACTIONS ON COMPUTER AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.