Back to the Future: Current-Mode Processor in the Era of Deeply Scaled CMOS

Yuxin Bai, Yanwei Song, Mahdi Nazm Bojnordi, Alexander Shapiro,
Eby G. Friedman, *Fellow, IEEE*, and Engin Ipek

***Abstract*— This paper explores the use of MOS current-mode logic (MCML) as a fast and low noise alternative to static CMOS circuits in microprocessors, thereby improving the performance, energy efficiency, and signal integrity of future computer systems. The power and ground noise generated by an MCML circuit is typically 10–100× smaller than the noise generated by a static CMOS circuit. Unlike static CMOS, whose dominant dynamic power is proportional to the frequency, MCML circuits dissipate a constant power independent of clock frequency. Although these traits make MCML highly energy efficient when operating at high speeds, the constant static power of MCML poses a challenge for a microarchitecture that operates at the modest clock rate and with a low activity factor. To address this challenge, a single-core microarchitecture for MCML is explored that exploits the *C*-slow retiming technique, and operates at a high frequency with low complexity to save energy. This design principle contrasts with the contemporary multicore design paradigm for static CMOS that relies on a large number of gates operating in parallel at the modest speeds. The proposed architecture generates 10–40× lower power and ground noise, and operates within 13% of the performance (i.e., 1/ExecutionTime) of a conventional, eight-core static CMOS processor while exhibiting 1.6× lower energy and 9% less area. Moreover, the operation of an MCML processor is robust under both systematic and random variations in transistor threshold voltage and effective channel length.**

***Index Terms*— Architecture-circuit codesign, energy efficient, low noise, microprocessors, MOS current-mode logic (MCML).**

I. Introduction

With technology scaling, signal integrity and power dissipation have become critical challenges that limit microprocessor performance. MOS current-mode logic (MCML) is a fast, low-noise differential logic style, which has garnered recent interest as a replacement for static CMOS circuits in noise-sensitive applications [1]–[3]. MCML is the CMOS successor of bipolar emitter-coupled logic (ECL), which has been used in high-speed applications since the 1970s. The Cray-1 [4], the IBM Enterprise System/9000 [5], and the IBM System/390 [6] all used ECL and differential current switching to achieve high speeds with low noise.[1]

MCML maintains the benefits of traditional ECL, such as high-speed, lower di/dt noise, and common-mode noise rejection, without relying on bipolar transistors [1]. It requires a constant current independent of frequency, making this technology particularly appropriate for high-speed integrated circuits (ICs). Unfortunately, MCML suffers from constant static power dissipation, which, if left unmanaged, would result in an inordinate energy requirement in large-scale ICs operating at moderate speeds (e.g., 1–2 GHz). This limitation heretofore has prevented the use of MCML as a replacement for static CMOS in mainstream multicore processors; however, signal integrity and power delivery challenges, as well as the increasingly large CMOS static power due to subthreshold leakage, are tilting the balance in favor of MCML—particularly in single-core processors operating at high frequencies.

This paper explores the use of MCML as an alternative to static CMOS circuits in throughput-oriented microprocessors (such as the Sun Niagara [8], [9], Oracle M7 [10], Intel Knights Landing [11], and Tilera TILEPro [12]), thereby improving the signal integrity, performance, and energy efficiency of future high-performance systems.[2] Mitigating static power requires exploiting the unique circuit-level power characteristics of MCML at the architecture level, breaking from the contemporary multicore design paradigm, which relies on a large number of gates operating in parallel at the modest speeds. A complexity-effective, deeply pipelined, and multithreaded architecture specific to the MCML is synthesized at 22 nm using an MCML standard cell library. Simulation results indicate that the proposed MCML processor achieves 10–40× lower power and ground noise, 1.6× lower energy, and 9% less area than a conventional, eight-core static CMOS system while operating within 13% of the performance (1/ExecutionTime) achieved by a static CMOS version. Moreover, the operation of the MCML processor is robust under both systematic and random variations in transistor threshold voltage and effective channel length.

II. Related Work

MCML was introduced by Mizuno *et al.* [13] in 1996 as a high-speed, low-noise, and analog friendly logic family for mixed-signal environments, and has been used in applications, such as CORDIC circuits, optical communication transceivers, high-speed ring oscillators, frequency dividers, and multiplexer

Manuscript received February 4, 2015; revised May 17, 2015; accepted June 26, 2015. This work was supported by the Directorate for Computer and Information Science and Engineering through the National Science Foundation CAREER under Award CCF-1054179.

The authors are with the Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627 USA (e-mail: yuxin.bai@rochester.edu; yanwei.song@rochester.edu; m.nazmbojnordi@rochester.edu; alexander.shapiro@rochester.edu; friedman@ece.rochester.edu; ipek@cs.rochester.edu).

Digital Object Identifier 10.1109/TVLSI.2015.2455874

[1]The low-noise environment has allowed the Cray-1 to use an entirely unregulated power supply [7].

[2]The MCML throughput cores can also be integrated with a conventional wide-issue static CMOS core within a heterogeneous multicore configuration.

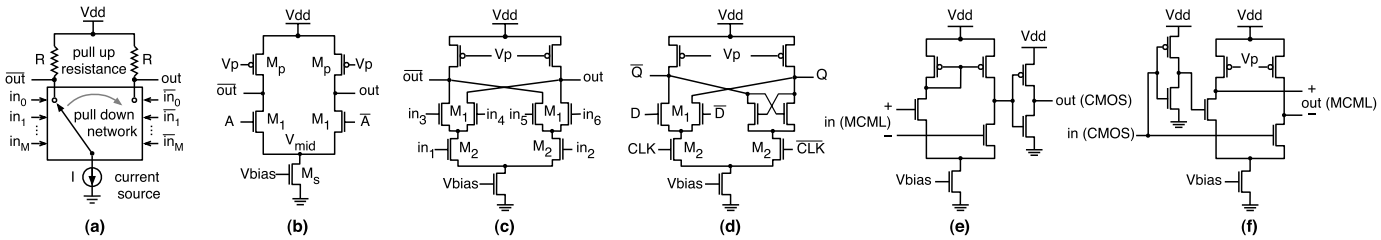


Fig. 1. Exemplary MCML gates and interface circuitry. (a) Ideal MCML gate. (b) Inverter/buffer. (c) Two-input universal gate. (d) D-latch. (e) MCML-to-CMOS conversion. (f) CMOS-to-MCML conversion.

circuits [1], [14]–[17]. Design guidelines are provided in [18]–[20] for MCML logic gates and MCML circuits under process variations. Badel [3] and Cevrero *et al.* [21] proposed design flows for optimizing MCML cell libraries at 0.18 um and 90 nm. Despite these advances, the high static power prevents MCML from wide adoption in large-scale digital circuits. Voltage noise poses major challenges in aggressive voltage scaling and reliable operation in static CMOS processors. Microarchitectural techniques have been developed to detect and control inductive noise [22]–[24], such as sizing and placing decoupling capacitors based on the expected level of microarchitecture activity [25]. Zhang *et al.* [26] and Gupta *et al.* [27] proposed Power Delivery Network modeling techniques to assess IR drops and di/dt noise. Software approaches in [28]–[32] aim at identifying, avoiding, and recovering from noise-induced voltage emergencies. While an issue in static CMOS processors, noise does not constitute a serious issue in MCML processors.

Simultaneous switching noise is a dominant component of noise [33]. Prior work explores power-gated [21] and dynamically controlled [34] MCML cells to support power/clock gating. The clock distribution of the MCML system uses MCML gates, which do not contribute significant power noise.

Power consumption and high complexity are key limiting factors that prevent deeply pipelined static CMOS processors (e.g., Intel Pentium 4 [35], [36]) from operating at an aggressive clock frequency. The dominant power component of an MCML processor is static and independent of frequency, which enables MCML to operate at a high frequency within a limited power budget.

Seok *et al.* [37] describe an Fast Fourier Transformation (FFT) core in the ultralow-voltage regime with superpipelining, thereby reducing leakage power. *C*-slow retiming has been applied to FPGA circuits, and automated tools have been developed [38], [39]. Various multithreading techniques have been applied to commercial processors, including simultaneous multithreading (SMT) [40] and chip multithreading [8]. These techniques inspire complexity-effective solutions for MCML processors.

III. Background and Motivation

The growing importance of signal integrity and power delivery challenges, compounded by the increasingly large CMOS static power due to subthreshold leakage current, are diminishing the advantages of static CMOS circuits over other logic families. This section provides an overview of the electrical noise and signal integrity issues in CMOS ICs, and the design principles behind MCML processors. Detailed description of MCML is beyond the scope of this paper; interested readers can find more information on MCML in [1], [3], [13], [20], and [41].

A. Signal Integrity Challenges in Deeply Scaled CMOS

Electrical noise can be categorized into device and switching noise. Device noise is intrinsic to transistors, including thermal, shot, and flicker noise. Switching noise results from the simultaneous switching of devices, which is a primary concern in the conventional CMOS digital circuits, and is typically two to three orders of magnitude greater than the device noise [42]. Switching noise is particularly critical in high-energy and high switching activity signal nets, such as clock networks [33]. The two primary coupling mechanisms for switching noise are: 1) power and ground noise and 2) interconnect noise. Power and ground noise is caused by high-frequency switching of the instantaneous current, which introduces large current spikes that IR and inductive (Ldi/dt) voltage drops, producing voltage fluctuations in the power and ground distribution networks [33], [43]–[46]. Interconnect noise [47]–[49] or crosstalk is the voltage change induced on a victim node due to capacitive or inductive coupling from a switching node.

CMOS digital circuits are traditionally considered tolerant to noise due to the high-noise margins [42]. This condition, however, has changed due to scaled power supply and threshold voltages. High-frequency signal transitions exacerbate the magnitude of the coupling noise, which makes signal integrity a significant challenge in high-speed digital ICs. As a result, microprocessor designers incorporate expensive hardware solutions and pessimistically employ the worst case design to ensure the reliable operation. Numerous microarchitecture techniques have been proposed in [22]–[32] and [50]–[53] to address inductive noise and IR drops at the expense of performance, power, and area. In mixed-signal circuits, the noise reduces the precision and the dynamic range of sensitive analog circuitry through capacitive and inductive coupling, resistive voltage drops, and threshold voltage variations [42]. The design of future microprocessors, therefore, requires innovative solutions to compensate for noise in highly scaled technologies.

B. MCML

An ideal MCML gate, as shown in Fig. 1(a), comprises three major parts: 1) a pair of pull-up load resistors; 2) a pull-down logic network; and 3) a constant current source. The pull-up

TABLE I
COMPARISON OF THE CMOS AND MCML GATE CHAINS FOR DELAY, POWER, ENERGY, AND ED PRODUCT. α AND k ARE PROCESS AND TRANSISTOR-SIZE-DEPENDENT PARAMETERS, C IS THE LOAD CAPACITANCE, I IS THE CONSTANT CURRENT, AND f IS THE FREQUENCY

	CMOS	MCML
Delay	$\frac{N\times C\times V_{dd}}{\frac{k}{2}(V_{dd}-V_{th})^{\alpha}}$	$\frac{N\times C\times V_{swing}}{I}$
Power	$N\times C\times V_{dd}^2\times f$	$N\times V_{dd}\times I$
Energy	$N\times C\times V_{dd}^2$	$N^2\times C\times V_{swing}\times V_{dd}$
ED product	$\frac{N^2\times C^2\times V_{dd}^3}{\frac{k}{2}(V_{dd}-V_{th})^{\alpha}}$	$\frac{N^3\times C^2\times V_{swing}^2\times V_{dd}}{I}$

load resistors are typically implemented with pMOS transistors, as shown in Fig. 1(b), for an MCML inverter. The pull-down network is fully differential, and generates both the true and the complementary forms of the output signal; consequently, logic can often be simplified by eliminating inverters. The constant current source can be implemented with a single nMOS transistor and typically uses a separate control voltage, V_{bias}. This constant current is steered between the differential branches (i.e., the pull up loads) to allow the outputs to change. The current drawn from the supply is usually much smaller and more stable than the CMOS switching current. This characteristic provides opportunities for MCML gates to generate far less switching noise than CMOS gates; it furthermore eases the constraints on the power delivery system, which makes it possible to reduce the design complexity and to lower the energy consumption of the system. However, the constant current causes high static power dissipation. The voltage swing for an MCML gate, $V_{\text{swing}} = I \times R$, can be adjusted by tuning the resistive loads and the tail current of the constant current source. It is usually not rail to rail and can be two to ten times lower than V_{dd}, which provides higher gate speeds.

Energy and Delay Characteristics: To compare MCML and CMOS gates, a set of basic properties is derived from a simple MCML model [20]. A comparison between the CMOS and the MCML gates in terms of delay, power, energy, and energy-delay (ED) product is listed in Table I. Assume a linear chain of N identical gates, each with a load capacitance C. The total propagation delay of this chain of gates is proportional to

$$D = N \times R \times C \tag{1}$$

where R is

$$R = \frac{V_{\text{swing}}}{I}. \tag{2}$$

For static CMOS, the current and voltage swing can be expressed as

$$I = \frac{k}{2} \times (V_{\text{dd}} - V_{\text{th}})^{\alpha}, \quad V_{\text{swing}} = V_{\text{dd}}. \tag{3}$$

The delay of static CMOS for an N-gate chain is, therefore

$$D_{\text{CMOS}} = \frac{N \times C \times V_{\text{dd}}}{\frac{k}{2} \times (V_{\text{dd}} - V_{\text{th}})^{\alpha}}. \tag{4}$$

Similarly, for MCML, the delay of an N-gate chain is

$$D_{\text{MCML}} = N \times R \times C = \frac{N \times C \times V_{\text{swing}}}{I}. \tag{5}$$

The dynamic energy for N static CMOS gates is $N \times C \times V_{\text{dd}}^2$, and the frequency f of a pipeline stage is constrained by $1/D_{\text{CMOS}}$. The static power of N MCML gates is $N \times V_{\text{dd}} \times I$. The energy and the power are listed in Table I.

One appealing property of MCML is that the gates do not exhibit a theoretically minimum ED product [1]. By adjusting the constant current within an MCML gate while maintaining the other parameters, the ED product can significantly be reduced (the limiting factor would be robustness). As suggested by the expressions listed in Table I, both the energy and the ED product benefit from the low-voltage swing, but the energy efficiency deteriorates with logic depth N (i.e., the number of gates connected in series in a single pipeline stage). It is, therefore, important for the MCML gates to maintain a low logic depth in each pipeline stage. This quality implies that the MCML circuits with constant power consumption can significantly be more energy efficient than the CMOS circuits (CMOS dynamic power increases with frequency) when operating at high speeds [1].

C. C-Slow Retiming: A Complexity-Effective Methodology for Designing Deep, Multithreaded Pipelines

C-slow retiming [38], [39] is a pipelining technique to improve the throughput of a circuit with feedback loops. It consists of an initial C-slow step, in which every register is replaced with a sequence of C registers, followed by a retiming step that balances the logic delay between the pipeline registers. In the ideal case, this optimization process improves the throughput of the pipeline by a factor of C while introducing additional latency for a single data stream. If C interleaved, independent data streams (e.g., threads) enter the pipeline on a round robin basis, no new combinational logic is needed to create new feedback loops between the stages. In this paper, C-slow retiming is combined with SMT to permit a digital processor to operate at high frequencies. In C-slow retiming, the performance and the power can both be improved by employing a low complexity circuit (superpipelining without C-slow retiming on an MCML processor would improve performance at the expense of increased complexity and static power). C-slow retiming is particularly important for MCML rather than static CMOS in terms of energy efficiency, since the power of MCML is static and independent of frequency.

Consider an in-order, five-stage-pipelined MIPS microprocessor, and assume that this pipeline is C-slow retimed. Each of the original pipeline stages is called a major pipeline stage, and each of the C new stages within a major stage is called a minor pipeline stage. After C-slow retiming, it takes C cycles to complete the work in one major stage. Independent instructions from different threads enter the pipeline in consecutive cycles, while instructions from the same thread are allowed to enter a major stage every C cycles. In other words, a time division multiplexing (TDM) fetch policy is adopted to enforce correct forwarding without any new bypassing paths

TABLE II
DESIGN SPACE EXPLORATION. VARIABLES WITH SUBSCRIPT 1 ARE FOR nMOS TRANSISTOR M_1 IN THE DIFFERENTIAL PAIR, SUBSCRIPT s FOR THE TAIL nMOS TRANSISTOR, AND SUBSCRIPT p FOR LOAD pMOS TRANSISTORS, AS SHOWN IN FIG. 1(b), IN THIS PAPER

Given variables: d_t, V_{swing}, W_1, L_1, W_p, L_p, W_s, L_s, I, V_p, V_{mid}, V_{bias}
Minimize: power consumption P
Subject to:
(1) Performance target, $t_d \leq d_t$
(2) Supply voltage, $0.4V \leq V_{dd} \leq 0.8V$
(3) Signal slope ratio, $SSR < 5$
(4) Voltage swing ratio, $VSR > 0.95$
(5) Within die variations (δ/μ), 9% for V_{th}, 4.5% for L_{eff}
(6) Gain and noise margin, $A_v > 2$, $NM > 0.4V_{swing}$
(7) Load resistance R, tuned with load PMOS transistors W_p, L_p, V_p
(8) Voltage swing, $V_{swing} = I \times R$, $0.05V \leq V_{swing} \leq V_{th_1}$
(9) Constant current, $I = W_s v_{sat} C_{ox}(V_{bias} - V_{th_s} - A_s V_{dsat_s})$
(10) Bias voltage, $V_{th_s} \leq V_{bias} \leq V_{mid} + V_{th_s}$
(11) Tail transistor, $V_{dd} - V_{th_1} - V_{swing} \leq V_{mid} \leq V_{dd} - V_{th_1}$

or control logic. The combination of C-slow retiming and TDM multithreading enables a deep, multithreaded pipeline with low complexity and few additional logic gates.

IV. FUNDAMENTAL BUILDING BLOCKS

In this section, the basic philosophy behind an MCML standard cell library and how this library differs from a CMOS standard cell library are described. Two example MCML gates (a two-input universal gate and a D-latch) and the conversion circuitry between the static CMOS and the MCML are discussed.

A. MCML Logic Gates

A 22-nm MCML standard cell library was developed to evaluate tradeoffs in MCML-based processors. The design space for each cell was explored by varying transistor sizes, voltage swing, output current, and $V_{\rm dd}$ [for detailed guidelines (see [20])]. Constraints obeyed in the design space exploration include: 1) tolerance to process variations (Section VII-A); 2) a sufficient voltage gain and noise margin to ensure stable signal regeneration; 3) a sufficient voltage swing ratio (defined as the ratio of current in the ON branch to the total current) for effective current steering; and 4) a signal slope ratio to maintain the transition time sufficiently short with respect to the propagation delay [54]. A detailed procedure is listed in Table II.

Stacking is restricted to at most two levels for the MCML gates to ensure robustness. This MCML library operates at 0.55 V with a 100 mV voltage swing, and is compared with two 22-nm static CMOS standard cell libraries: one operating at 0.8 V (nominal supply voltage at 22 nm) and one operating at 0.55 V—the same supply voltage as the MCML library. Delay, area, and power comparisons for seven representative cells are listed in Table III. The differential operation of MCML requires in 2× more transistors and higher wiring overhead in the inverter and NAND2 gates. However, for XOR2 and MUX2 gates, the number of transistors in the MCML version is similar to that of CMOS with comparable area. MCML gates dissipate mostly static power (microwatt), while CMOS gates consume dynamic power proportional to the switching frequency (microwatt/gigahertz).

1) Universal Gates: A two-input universal gate [Fig. 1(c)] can implement eight logic functions (AND, OR, XOR, MUX, and complementary forms) by exchanging input signals. Different input combinations are listed in Table IV to construct various logic functions with the same gate. Three versions of this universal gate with different drive strengths are incorporated into the MCML library (capable of emulating 19 gates in the CMOS library).

2) Storage Structures: In an MCML processor, a natural way of implementing storage structures is to use MCML-based latches and flip-flops. Fig. 1(d) shows an MCML-based D-latch. The highest level differential pairs function differently. One of the pairs serves as a sample branch to sense the input data, while the other cross-coupled pair serves as a hold branch to store the data, as shown in Fig. 1(d). When the clock is high, the current passes through the sample branch (left) and changes the output to the inverted value of the input data. When the clock is low, the hold branch is activated, and the cross-coupled transistors with regenerative positive feedback maintain the output in the same state. A simple MCML master-slave D flip-flop (DFF) can be constructed by connecting two of these D-latches. These storage cells maintain the low-noise benefits of MCML, but suffer from constant static power dissipation. This power is tolerable in DFFs in small structures with a high activity factor (such as a pipeline register), but is a problem for larger storage structures (such as L1 or L2 caches) with a low activity factor per cell. To reduce static energy, the proposed architecture uses SRAM cells for L1 and L2 caches, register files, and Translation Lookaside Buffers (TLBs) and provides the appropriate support for interfacing between the MCML and the SRAM domains.

B. MCML-Static CMOS Interface

Interfacing an MCML-based processor core to SRAM-based storage structures requires signals to be translated between a low-swing differential voltage domain and a full-swing single-ended voltage domain. Conversion from MCML to static CMOS requires a differential to single-ended amplifier with low gain (usually no more than ten) plus a static CMOS inverter that amplifies the low-voltage swing beyond the switching threshold of a CMOS gate. For a small output load, the amplifier can be fast despite the small source current [55], as shown in Fig. 1(e). Voltage conversion from the CMOS to the MCML is simpler: the CMOS voltage swings are larger than the MCML, and the MCML gates can operate at a higher voltage swing than the target operating point. Inserting a single CMOS inverter to generate a complementary signal and a single MCML inverter to balance the two signals is sufficient to interface CMOS to MCML [Fig. 1(f)]. Area and power overheads depend upon the total number of signals that cross the boundary between the static CMOS and the MCML modules, which is not significant according to the evaluation.

V. DESIGN PHILOSOPHY

Power dissipation in MCML circuits is roughly proportional to the number of logic gates, but is independent of the switching frequency. As a result, the intuition behind an energy-efficient, current-mode microprocessor is: 1) to avoid using a large number of gates and 2) to maintain a shallow

TABLE III

DELAY, AREA, AND POWER COMPARISON OF MCML AND STATIC CMOS (0.8 AND 0.55 V) STANDARD CELL LIBRARIES. ASSUME A 100% SWITCHING ACTIVITY FOR STATIC CMOS CELLS WHEN OPERATING AT A 1 GHz, 17 ps INPUT TRANSITION TIME, 1 fF LOAD CAPACITANCE,[3] AND THE SAME INPUT SWITCHING PATTERNS FOR ALL OF THE CELLS

Basic Cells	Delay			Area			Power		
	MCML (ps)	CMOS0.8 (ps)	CMOS0.55 (ps)	MCML (μm^2)	CMOS0.8 (μm^2)	CMOS0.55 (μm^2)	MCML (μW)	CMOS0.8 (μW)	CMOS0.55 (μW)
INVX1	8.50	7.70	22.70	0.67	0.34	0.34	1.05	0.52	0.29
NAND2X1	10.10	10.70	40.77	1.30	0.45	0.45	0.89	0.77	0.31
NOR2X1	17.25	13.30	53.11	1.30	0.56	0.56	0.89	0.84	0.38
XOR2X1	20.02	18.42	54.60	1.30	1.22	1.22	1.51	1.67	0.73
MUX2X1	15.20	19.01	51.38	1.30	1.19	1.19	1.51	1.34	0.54
DFFX1	49.00	34.01	129.01	3.12	1.91	1.91	4.11	2.23	0.84
LATCHX1	34.02	27.17	92.05	1.59	1.23	1.23	1.56	1.65	0.55

TABLE IV

INPUT CONNECTIONS FOR A TWO-INPUT UNIVERSAL GATE TO CONSTRUCT DIFFERENT LOGIC FUNCTIONS

Logic function	in_1	in_2	in_3	in_4	in_5	in_6
XOR/NOXR	$\overline{A}$	A	B	$\overline{B}$	$\overline{B}$	B
MUX/NMUX	S	$\overline{S}$	D_1	$\overline{D_1}$	D_0	$\overline{D_0}$
AND/NAND	A	$\overline{A}$	B	$\overline{B}$	A	$\overline{A}$
OR/NOR	B	$\overline{B}$	B	$\overline{B}$	A	$\overline{A}$

logic depth in each pipeline stage to minimize the time each gate remains idle within a clock period. Maintaining energy efficiency in a current-mode processor, therefore, requires a complexity-effective design that operates at a high frequency to reduce execution time and static energy. A significant challenge that often plagues a deeply pipelined datapath is the problem of overcoming data dependence to keep the pipeline highly utilized. A secondary problem that is particularly important in the case of MCML is the static power overhead of the extra pipeline latches and the control logic, which can offset the energy-efficiency benefits of a deep pipeline. To make a high-speed, deep pipeline viable for a current-mode processor, two synergistic techniques are explored: 1) C-slow retiming and 2) multithreading. Recall that, in C-slow retiming (Section III-C), each of the original pipeline stages (except writeback) in a baseline circuit is partitioned into C stages. The resulting pipeline ($4C + 1$ stages) works correctly without any new forwarding paths or control logic as long as the threads are interleaved, and the instructions from a given thread are restricted to issuing every C cycles.

Choosing the Optimum C: A critical parameter that determines the energy efficiency of the proposed C-slow MCML processor is C. It is possible to derive a simple analytic model that expresses system energy as a function of C and other design parameters to understand the tradeoffs involved in optimally selecting C.

Assume that a baseline MCML processor has a static power dissipation of P_0, and the static power of a C-slow pipeline is P_c. Let the marginal increase in static power (extra MCML latch power) from a C-slow circuit to a $(C + 1)$-slow circuit be P_δ for the entire pipeline; thus

$$P_c = P_0 + (C^\beta - 1)P_\delta \tag{6}$$

where the factor β describes the increasing power overhead with increasing pipeline depth (e.g., linear when $\beta = 1$ or superlinear when $\beta > 1$) [56], [57]. The critical path delay is composed of two elements: 1) the combinational delay d_0 and 2) the pipeline latch delay d_δ. Since the combinational portion is composed of C stages by C-slow retiming, the critical path delay of the new pipeline stage, d_c, is $d_0/C + d_\delta$. The static energy consumed in one cycle by the C-slow pipeline, E_c, is

$$E_c = P_c d_c = (P_0 + (C^\beta - 1)P_\delta)\left(\frac{d_0}{C} + d_\delta\right). \tag{7}$$

For example, if the latch overhead increases linearly with C ($\beta = 1$), the energy can be expressed as

$$E_c = \frac{(P_0 - P_\delta)d_0}{C} + C P_\delta d_\delta + (P_0 d_\delta + P_\delta d_0 - P_\delta d_\delta). \tag{8}$$

This expression indicates that the energy to run an application on a C-slow MCML pipeline depends on two competing factors: 1) the lower static energy wasted in each pipeline stage as C is increased and 2) the lower energy overhead of latches as C is decreased. Setting the derivative of E_c (with respect to C) to zero yields the optimum C and C_{opt} that minimizes static energy per cycle

$$C_{\text{opt}} = \sqrt{\left(\frac{P_0}{P_\delta} - 1\right)\frac{d_0}{d_\delta}}. \tag{9}$$

When the latch overhead (P_δ and d_δ) is negligible as compared with the baseline pipeline (P_0 and d_0), C_{opt} is large; in this regime, the deeper the pipeline, the lower the energy. Practical limitations on the achievable energy efficiency are the maximum reachable pipeline depth and the available thread-level parallelism. If the latch overhead is comparable with the energy of the baseline pipeline; however, the energy efficiency decreases beyond provided by C_{opt}. For example, when $P_0 \approx 8P_\delta$ and $d_0 \approx 9d_\delta$, C_{opt} is 8, indicating that a 33-stage ($4C + 1$) pipeline is the most energy efficient circuit. In the case, where the latch overhead increases superlinearly (i.e., β is >1), the optimum C is smaller. The energy efficiency is, therefore, primarily constrained by the pipeline overhead and the required robustness.

[3]In an actual case during synthesis, a range of delays is dynamically determined by the synthesis tool; the total load capacitance is typically smaller, and the input transitions are faster after the synthesis process. In the deepest evaluated pipeline (8*C* 32 T), a stage has exhibited a delay of about nine to ten FO4s (Section VII-C).

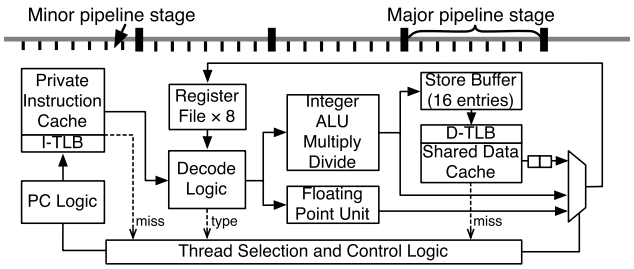


Fig. 2. Proposed C-slow ($C = 8$) pipeline structure for the current-mode processor.

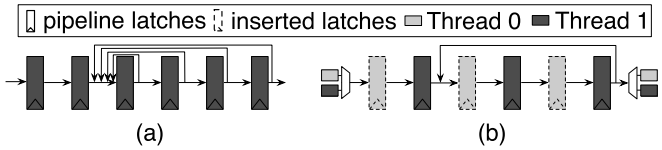


Fig. 3. Scheduling multiple threads in a conventional deep pipeline and a two-slow pipeline. (a) Consecutive instructions from the same thread require a full set of forwarding paths. (b) Instructions from interleaved threads use a simple forwarding path.

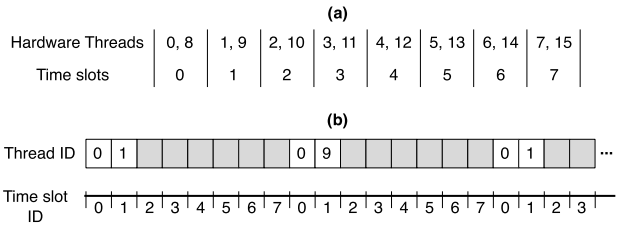


Fig. 4. Instruction assignment to time slots. (a) Assignment of the 16 threads to $C = 8$ time slots. (b) Executing instructions for active threads (0, 1, and 9).

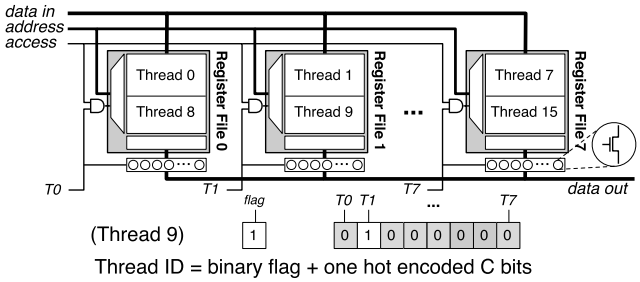


Fig. 5. Illustrative example of accessing the register files (eight-slow core with 16 hardware threads). T0–T7: eight time slot enable signals.

VI. Pipeline Organization

The proposed complexity-effective current-mode microprocessor is a C-slow version of a single-issue, in-order, and multithreaded core with five stages. Each of the fetch, decode, execute, memory access, and writeback stages constitutes a major pipeline stage, and each of the four major pipeline registers is replaced with C pipeline registers using C-slow retiming. This process converts the baseline core into a deeply pipelined processor with $4C$ pipeline registers and $4C + 1$ stages (e.g., 33 stages in Fig. 2). To avoid the need for extra forwarding and control logic, a restriction is placed on when a thread can enter a major pipeline stage. In particular, each clock cycle is assigned a time slot identifier (ID) between 0 and $C - 1$ to ensure that a clock cycle with a time slot ID of t is followed by a clock cycle with a time slot ID of $(t + 1)$ mod C. Each thread is statically assigned to a time slot. For example, if $C = 8$ and a thread is assigned to time slot 1 that thread is eligible to enter the pipeline on cycles 1, 9, 17, 25, and so on. Interleaving the threads obviates the need for extra forwarding paths. From the point of view of a single thread, the processor is logically equivalent to the baseline five-stage pipeline running at $1/C$ of the actual clock rate. This concept is shown in Fig. 3(b) and is compared with a conventional, more complex pipeline with extra forwarding paths in Fig. 3(a).

A. Instruction Fetch

The instruction fetch stage implements a fair fetch policy that allocates threads to each time slot in a round-robin fashion. Each time slot is identified by a time slot ID [Fig. 4(a)], and assigned a predesignated set of hardware thread contexts. Fig. 4 shows how threads are interleaved. In this example, threads 0, 1, and 9 are available for execution, while other threads are stalled for memory. Thread 0 is statically assigned to time slot 0, while threads 1 and 9 are assigned to time slot 1. Threads 1 and 9, therefore, take turns sharing time slot 1. The rest of the slots carry no operations, waiting for one of the assigned threads to become available.

1) Program Counters: Each hardware thread is assigned a 32-bit program counter (PC). Given T hardware threads, a set of T/C PC registers is statically allocated to each time slot. The fetch unit selects one of the ready PCs every cycle using a two-level round-robin selection policy. The time slot ID is initially used to select the corresponding set of PCs in round-robin order; next, within the selected set, one of the ready threads is selected (also in round-robin order).

2) Instruction Cache: The instruction cache consists of C 16-KB cache banks. Each cache bank is dedicated to a particular time slot and provides a throughput of one access every C cycles. All threads assigned to the same time slot share the same cache bank. Since a bank can only be accessed in a dedicated time slot (i.e., every C cycles), there is never a bank conflict among threads. One 32-entry, four-way set associative instruction TLB is used for each of the C cache banks.

B. Instruction Decode

The decode stage accesses both the instruction decoder and the register files.

1) Decoder: The decoder implements the MIPS instruction set architecture. It detects hazards and updates the state of the current thread. Since the architecture guarantees that a thread is not assigned to more than one time slot and the original five-stage pipeline handles the necessary operand bypassing, no new hazards exist in the C-slow pipeline.

2) Register Files: The processor incorporates C separate register files, each of which is dedicated to one time slot. If the number of hardware threads is T, every register file is partitioned into T/C banks that share two read ports and a write port. As shown in Fig. 5, 16 hardware threads are assigned to eight time slots. Each register file is enabled every

eight cycles through shared input and output buses. Two banks of the same register file (e.g., the banks for thread 0 and thread 8 in register file 0) are never accessed within the same eight-cycle period, and share the same read and write ports. This process eliminates the need for extra register file ports, large selection logic, and data routing for the inputs and outputs between the register file banks.

C. Execute

The execute stage includes an integer ALU, an FPU, a branch resolution unit, and the forwarding logic.

1) Integer ALU: The integer ALU consists of a logic unit, an adder/subtracter, a shifter, and a multiplier/divider. The multiplier/divider implements a 32-stage Booth algorithm, which allows for a high clock rate with a relatively small number of gates. Recall that, for an MCML circuit, a smaller gate count translates to lower static power; hence, a small, highly utilized unit delivers higher energy efficiency. The multiplier/divider is C-slow retimed, which results in a latency of $32C$ clock cycles. Since the path is pipelined, consecutive multiply and divide instructions from different threads as well as independent multiply/divide instructions from the same thread do not need to wait. However, if the next instruction from the same thread is not a multiply or a divide, this thread is blocked to prevent out-of-order completion. This event results in a thread switch within the corresponding time slot.

2) Floating Point Unit: The FPU is based on the open source OR1K floating point unit [58], and implements a four-stage, IEEE 754 compliant, single precision pipeline. Each original FPU stage requires two baseline clock cycles ($2C$ cycles in the C-slow pipeline); this time amounts to $8C$ cycles for conversion, addition, and subtraction operations. For multiply and divide, a 32-stage Booth algorithm is used to minimize gate count; after C-slow retiming, this translates to $38C$ cycles of delay.

3) Branch Resolution Unit: Branches are resolved in the execution stage, which requires nullifying one over-fetched instruction on a taken branch. This strategy results in a shorter critical path, and consequently a higher frequency and better overall performance than resolving branches in the decode stage of the proposed architecture.

D. Data Memory Access

The memory stage implements load and store instructions by interfacing to a shared store buffer and a shared L1 data cache. The L1 data cache comprises C cache banks, each supported by a 32-entry, four-way set associative data TLB. A memory operation is initiated in a single cycle, and completes within C cycles on a cache hit. A data TLB or data cache miss results in a context switch to another thread assigned to the same time slot. Since L1 data cache banks are shared, there exists a possibility of bank conflicts among different threads when the clock period is shorter than the cache cycle time, which can happen when C is large. When a bank conflict is detected, the entire pipeline freezes for typically one or two minor cycles until the conflicting bank becomes available (the impact of bank conflicts was investigated and found to not constitute a performance bottleneck; the conflict rate is 6.4% when $C = 8$). After the conflict is resolved, the blocked request is sent to the data cache bank and the pipeline restarts.

Store Buffer: The load-store unit implements a store buffer with one entry per hardware context. By checking the physical tags in the store buffer, a read-after-write hazard between the loads and the stores can be detected, and resolved by store–load forwarding.

E. Register Writeback

If necessary, the retiring thread writes the result to the register file in the writeback stage. The time slot ID and the thread ID determine, respectively, the destination register file and the corresponding bank.

TABLE V
TOOL CHAIN USED IN THE EVALUATION

Task	Tool
Cell design & noise simulation	22nm Predictive Technology Model [59], Cadence Virtuoso 6.1.5
Synthesis	Synopsys Design Compiler F-2011.09
RTL & netlist simulation	Verilator [60], Cadence NCSim [61]
C-Slow retiming	Customized C-slow retiming application in Perl
Power evaluation	Synopsys PrimeTimePT & CACTI 6.5 [62]
Place & Route	Cadence Encounter 9.1
Architectural simulation	SESC Simulator [63]

VII. EXPERIMENTAL SETUP

Assessing the performance, energy, area, and signal integrity characteristics of an MCML processor requires both the architectural and circuit-level design and the evaluation. The experimental setup used in the design of the MCML gates, RTL implementation, and architectural simulations are described in this section. Table V gives an overview of the tool chain used in the evaluation.

A. Standard Cell Library

The MCML cells use a 0.55 V power supply, which is the most energy-efficient voltage level from 0.8 to 0.4 V based on the experimental results. First, a 22-nm CMOS cell library operating at the nominal voltage (0.8 V) is implemented by scaling the 45-nm FreePDK [64] standard cell library to 22 nm. Next, a low-voltage CMOS library operating at 0.55 V is used to compare MCML with CMOS at the same operating voltage.

Since variability is a key concern in deeply scaled technologies, MCML cells are evaluated under process variations [20], [65]. A $\pm 12\%$ die-to-die threshold voltage variation is assumed [66], and four process corners (fast–fast, fast–slow, slow–fast, and slow–slow) are evaluated for both the CMOS and the MCML processors. An important difference between the static CMOS and the MCML gates is that MCML is more sensitive to transistor mismatch. The universal gate, which employs a symmetric differential style, is chosen to implement most logic gates in the proposed MCML-based system while considering structural balance and possible branch mismatch. To consider the impact of mismatch

between the differential branches of an MCML gate, a within-die random threshold voltage variation (σ/μ) of 6.3%, and a within-die random effective channel length variation of 3.2% are assumed. Within-die systematic variation is assumed independent and equal to the random variation, which results in total within-die variations (σ/μ) of 9% $(\sigma_{\text{rand}}^2 + \sigma_{\text{sys}}^2)^{1/2}$ and 4.5% for threshold voltage and effective channel length, respectively [for model details (see [67])]. To tolerate up to 6σ within-die variation, basic gates are upsized according to the well-known $\sigma \propto 1/(\sqrt{\text{area}})$ law [68]. Delay distributions due to parameter variations are derived from Monte Carlo simulations (SPICE) for each type of gate. The target processor is synthesized 1000×; each time, the synthesis tool statistically evaluates the critical path delay based on the cell delay distribution in the library to determine the delay distribution of the critical path of the processor (Section VIII-F).

B. RTL Design

A five-stage, in-order MIPS processor ($C = 1$) was designed in Verilog RTL, and synthesized with the three libraries as a baseline. The C-slow netlist ($C = 2$, 4, and 8) was retimed with the design compiler [69] to achieve a minimum clock period, and simulated and verified by NCSim [61]. A fixed clock uncertainty (corresponding to clock skew and jitter), which constrains the achievable frequency, is specified during logic synthesis. In addition, high-performance microprocessors utilize a clock mesh to minimize the clock skew by trading off power dissipation. The mesh averages the arrival times for adjacent nodes by balancing the current flowing through the mesh [70], [71]. The energy and the power of a static CMOS pipeline are determined from the netlist simulation, and synthesis using PrimeTime [72]. The power of the MCML pipeline is determined from the static power of each type of gate and the gate count (including the MCML–CMOS interface gates). All the SRAM structures for both the static CMOS and the MCML processors use the same SRAM cells, and operate at the nominal voltage (0.8 V); the energy and the power estimates are calculated using CACTI 6.5 [62].

Due to the differential routing requirements of MCML, a fair area comparison between the CMOS and the MCML requires a physical layout. Similar to [73] and [74], a fat-wire technique is adopted to consider the differential routing overhead of MCML (this technique doubles the pitch width of the wires, and supports minimal spacing rules). The 22-nm core area for the CMOS and the MCML is scaled from the 45-nm layout area.[4]

C. Noise Model

Three single-core processors, implemented with 0.8 V CMOS, 0.55 V CMOS, and 0.55 V MCML libraries, are simulated for the noise evaluation. The system-level noise is evaluated using a methodology similar to [75]–[78]. The noise model decouples the linear (power and ground network), and nonlinear (transistors) portions of the system, as shown in Fig. 6. The power and ground networks are modeled as resistive networks with parasitic capacitors, and the nonlinear switching blocks (e.g., decode and execute stages) are modeled as distributed current generators connected in parallel between the power and the ground networks. These current generators are implemented with representative circuits (e.g., 32 bit adders, decoders) rather than ideal current sources, and the current of each generator is proportional to the power consumption of the relevant block [75].

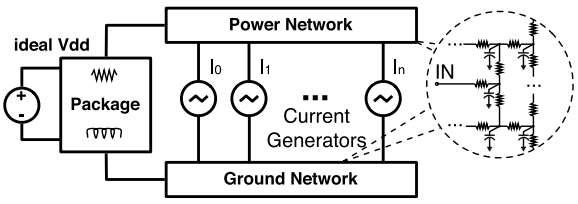


Fig. 6. Model for the system-level noise evaluation.

The metrics of each cell include the peak-to-peak power/ground voltage and the current noise. At the system level, a noise model (see Fig. 6) mimics the primary components in the noise generation process. The total capacitance, including interconnect and decoupling capacitors on the power and ground networks, is 0.4 pF/1000 gates.

D. Architecture

A modified version of the SESC simulator [63] is used to model both the static CMOS and the MCML processors. To explore the design space, several design choices (multicore versus C-slow, optimal number of cores and hardware thread contexts, and optimal frequency and supply voltage) are evaluated for both the CMOS and the MCML.[5] Note that the best design choice for the static CMOS is multicore instead of C-slow, while the opposite is true for MCML. As listed in Table VI, a C-slow ($C = 1, 2, 4, 8$), a single-core MCML processor is compared with a 2 GHz C-core CMOS (0.8 V) processor. Both the processors support a maximum of $T = 64$ hardware threads. The total cache capacity, number of threads, and other per-thread resources remain the same for the CMOS and the MCML for a given C. The frequency of the 0.55 V static CMOS processor is set to 667 MHz based on synthesis results; the experimental setup for the 0.55 V CMOS is otherwise identical to 0.8 V CMOS.

E. Applications

A mix of fourteen scalable parallel applications from the Phoenix [81], SPLASH-2 [82], SPEC OpenMP [83], and NAS [84] suites, as well as a batch self-organizing map (BSOM) application [85] are evaluated, as listed in Table VII.

VIII. Evaluation

This section compares the MCML and CMOS processors in terms of signal integrity, performance, energy, and area.

[4]FreePDK provides the design rules at a 45-nm technology, but not 32 nm or 22 nm.

[5]C-slow multithreading was applied to the CMOS C-slow circuits during the design space exploration.

TABLE VI
EXPERIMENTAL SETUP FOR STATIC CMOS (0.8 V) AND MCML-BASED SYSTEMS

	Static CMOS	**MCML (single core)**	**L1 memory** — direct-mapped instruction cache (i), 4-way set associative data cache (d)
1C	1 core @2GHz	5-stage @2GHz, 1 time slot	16KB i-L1, 16KB d-L1 per core
2C	2 cores @2GHz	9-stage @3.7GHz, 2 time slots	CMOS: 16KB i-L1, 16KB d-L1 per core; MESI coherence MCML: 16KB private i-L1 bank per time slot; two shared 16KB d-L1 banks
4C	4 cores @2GHz	17-stage @6.8GHz, 4 time slots	CMOS: 16KB i-L1, 16KB d-L1 per core; MESI coherence MCML: 16KB private i-l1 bank per time slot; four shared 16KB d-L1 banks
8C	8 cores @2GHz	33-stage @13GHz, 8 time slots	CMOS: 16KB i-L1, 16KB d-L1 per core; MESI coherence MCML: 16KB private i-L1 bank per time slot; eight shared 16KB d-L1 banks
L2 cache	8MB, 16-way set associative, 8 banks, 64-byte cache block size throughout all caches		
Memory	DDR2-1066 [64], 2 channels, 1 rank/channel, 8 banks/rank, FR-FCFS scheduling [80]		

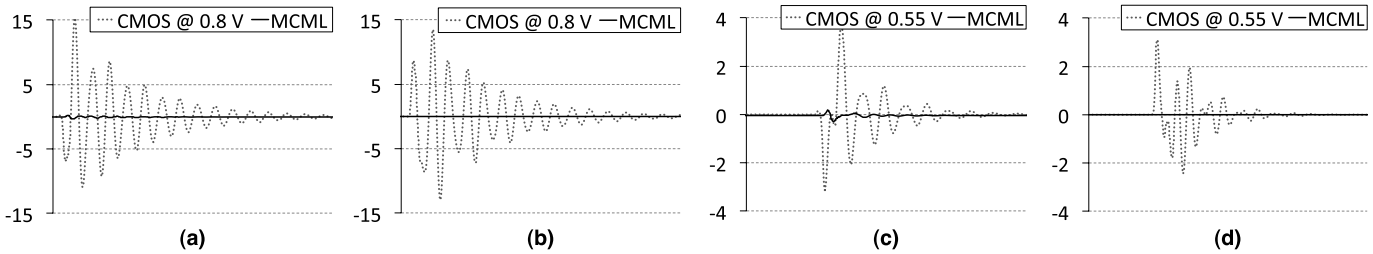


Fig. 7. Voltage noise of a single-core system implemented in the static CMOS (0.8 and 0.55 V) and the MCML. The y-axis is the noise percentage normalized to the supply voltage. (a) Power network noise. (b) Ground network noise. (c) Power network noise. (d) Ground network noise.

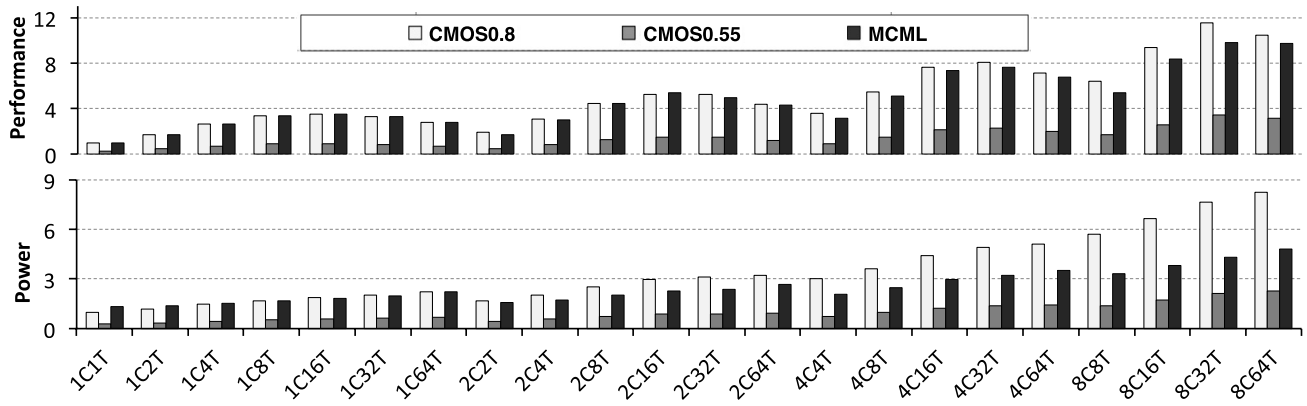


Fig. 8. Performance and power comparison among CMOS 0.8, CMOS 0.55, and MCML processors. 2*C* 32 T: two-core 32-thread for CMOS, and two-slow 32-thread single core for MCML. All data are normalized to CMOS 0.8 V 1*C* 1 T.

TABLE VII
APPLICATIONS AND DATA SETS

Benchmarks	**Suite**	**Input**
BSOM	NA	census
Histogram	Phoenix	small
Linear Regression	Phoenix	50MB key file
CG	NAS OpenMP	Class A
FT	NAS OpenMP	Class S
MG	NAS OpenMP	Class A
Swim-Omp	SPEC OpenMP	MinneSpec-Large
Equake-Omp	SPEC OpenMP	MinneSpec-Large
Barnes	SPLASH-2	16K Particles
Cholesky	SPLASH-2	tk 15.0
FFT	SPLASH-2	1M points
LU	SPLASH-2	512×512 matrix, 16×16 blocks
Ocean	SPLASH-2	514×514 ocean
Water-NSquared	SPLASH-2	512 molecules

The impact of process variations and the availability of thread-level parallelism on the performance and the power are also explored.

A. Signal Integrity

A full-swing static CMOS processor generates significant switching noise that is transferred from the digital blocks to the system through the power and ground networks, substrate, and interconnect. In contrast, the MCML processor draws a nearly constant current, which exhibits small fluctuations due to charging and discharging parasitic junction capacitances. A single static CMOS core at the nominal voltage (0.8 V) generates a 40× larger peak-to-peak voltage noise on the power and ground networks than an MCML core with comparable performance [Fig. 7(a) and (b)]. A slow low-voltage CMOS core (0.55 V) generates 10× larger peak-to-peak voltage noise than the MCML core [Fig. 7(c) and (d)], exhibiting 3× lower performance.

B. Performance and Energy

Fig. 8 shows the performance (1/execution time) and average power over 14 benchmarks operating on 0.8 V CMOS multicore, 0.55 V CMOS multicore, and MCML C-slow processors. T is the number of hardware threads, while C is the number of cores for CMOS and the C-slow factor for MCML. Since the clock frequency is constrained by the supply voltage in CMOS, the 0.55 V CMOS processor

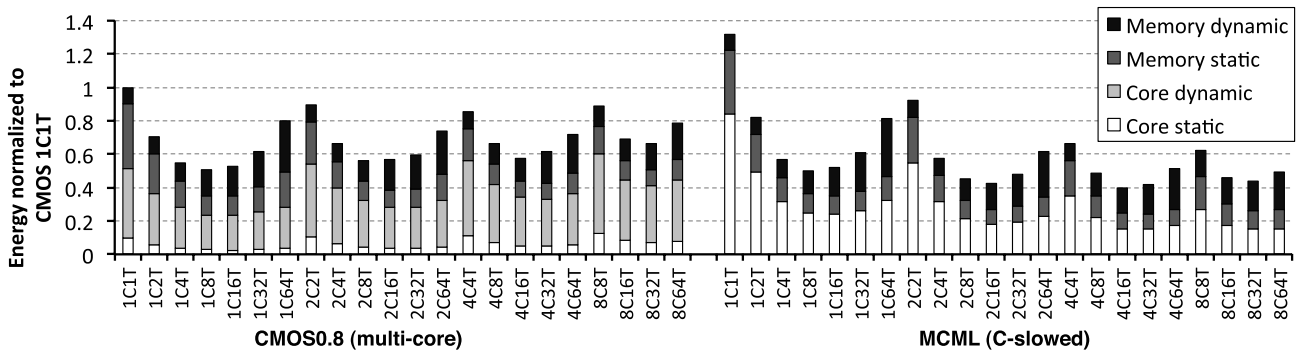


Fig. 9. Energy breakdown for the CMOS 0.8 V multicore and the MCML C-slow single-core processors. The energy for register files and TLBs are included in the memory energy.

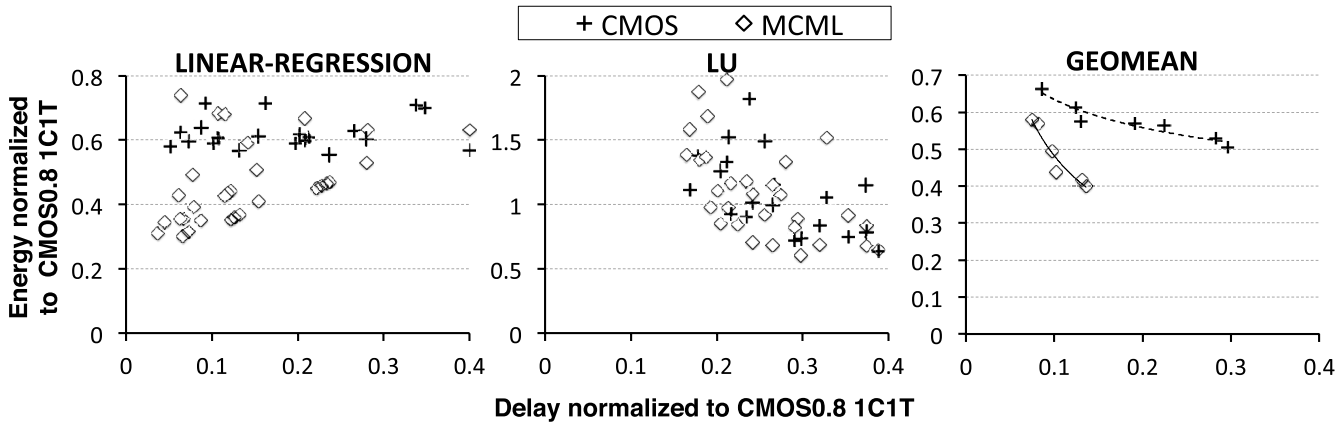


Fig. 10. Energy and Delay Pareto plots for representative benchmarks. Design points, including single core, multicore, multiple C-slow cores, and single C-slow core are presented for both the CMOS (0.8 and 0.55 V) and the MCML. Points beyond the scales are truncated. Geometric mean plot: Pareto frontiers of CMOS and MCML design points.

operates at 667 MHz, which results in lower performance than the 0.8 V CMOS and the MCML processors. MCML gates are generally faster than the static CMOS gates operating at the same supply voltage, and match the speed of static CMOS at nominal supply voltages. The MCML core delivers better power efficiency (up to 1.78×) than the 0.8 V CMOS multicore system when $C \geq 2$, because the power overhead of increasing pipeline depth in the C-slow MCML processor is lower than increasing the number of cores in the CMOS processor.

Fig. 9 shows the energy breakdown for the C-slow MCML and 0.8 V CMOS multicore processors. The MCML dynamic core energy is not significant, because the MCML core consumes nearly constant power regardless of the switching activity. The fastest MCML processor (8C 32 T) consumes 2.7× lower energy in the cores, and 1.6× lower energy in the system as compared with the fastest static CMOS processor (8C 32 T) at 0.8 V. The energy in the memory subsystem is over half of the total energy in the MCML processor, which is a bottleneck to achieving larger energy gains. Since the shared L1 data cache of MCML can suffer from conflicts, the dynamic memory energy of MCML is slightly higher than the corresponding CMOS processor. The small difference in static memory energy is due to the different execution times. The interface gates operate in parallel and negligibly increase the critical path delay. The total power and the area overhead of all of the interface circuits are, respectively, 3.7 mW and 0.0036 mm^2.

C. Exploration of Design Space

In this section, the design space of CMOS and MCML processors are explored to determine Pareto optimal configurations. The design space considers four design parameters: 1) supply voltage (0.8 and 0.55 V for CMOS and 0.55 V for MCML); 2) C-slow factor ($C = 1$, 2, 4, or 8); 3) number of cores (one, two, four, or eight); and 4) number of hardware threads (up to 64). Fig. 10 shows the system-wide ED for both the static CMOS and the MCML circuits. The individual benchmarks can roughly be categorized into two groups depending upon whether the MCML has a superior Pareto frontier than the CMOS. For clarity and space considerations, Fig. 10 only shows two representative benchmarks—LINEAR-REGRESSION and LU—from each group, as well as the geometric mean of all 14 benchmarks. As shown in the geometric mean plot, MCML processors generally exhibit higher performance and lower energy than the CMOS counterparts. The 0.8 and 0.55 V multicore circuit is constitute the boundary on the CMOS Pareto curve instead of the C-slow CMOS circuits, since the increased frequency and the pipeline overheads of C-slow retiming

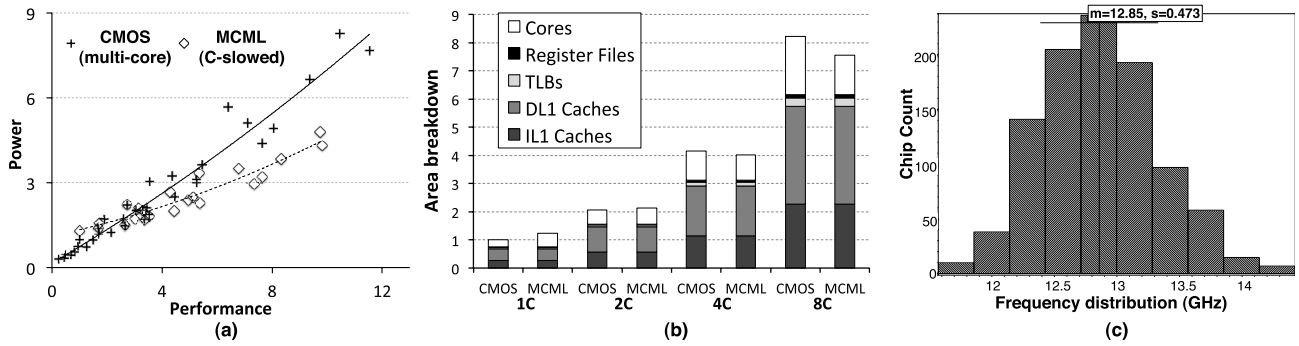


Fig. 11. (a) Power crossing. (b) Area comparison between the 0.8 V CMOS multicore and the MCML C-slow designs. (c) Monte Carlo simulation of within-die variations for 1000 MCML chips (eight-slow).

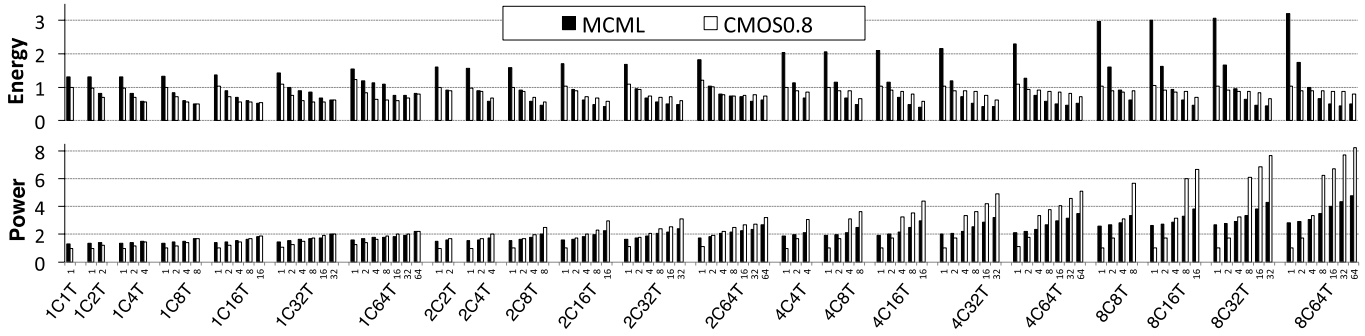


Fig. 12. Energy and power comparison of the CMOS and MCML processors when hardware threads are underutilized. 2C: two cores for CMOS, and a single two-slow core for MCML. 32 T: 32 available hardware threads. The numerical value under each column shows the number of software threads running on the system. All data are normalized to the 0.8 V 1C 1 T CMOS baseline. Idle cores of the 0.8 V CMOS processors are powered OFF.

significantly increase the power consumption of the static CMOS circuits (note that C-slow can be considered a form of superpipelining). For MCML, C-slow circuits are better than multicore circuits in energy, because the static power overhead from increasing the number of cores is larger than increasing the pipeline depth. Since MCML power dissipation is mostly static, reducing the execution time directly leads to energy savings in scalable parallel applications, such as BARNES, BSOM, CG, FFT, LINEAR, OCEAN, SWIM, and WATER. In LINEAR-REGRESSION, the static CMOS multicore processor suffers from false sharing in the memory subsystem, resulting in lower energy efficiency than the single-core MCML processor. However, in applications, such as LU, EQUAKE, FT, CHOLESKY, HISTOGRAM, and MG, C-slow MCML does not offer a clear energy advantage because of the relatively weak performance due to the high bank conflict rate (LU), and high miss rate (CHOLESKY and MG) in the shared L1 data cache (not the case for the CMOS processor, which has a private L1 data cache per core).

Due to process variations and clock skew, however, it is challenging to further deepen the C-slow pipeline: the eight-slow processor achieves a clock frequency of 13 GHz, which corresponds to nine to ten FO4s. By incorporating a small number of C-slow cores, MCML circuits (dual-core eight-slow, 32 and 64 T) outperform the fastest CMOS processor (8C 32 T) by 16%, and achieve energy savings of 19% (the top two points on the GEOMEAN).

A crossover point in the design space separates the regions where either MCML or CMOS is more power efficient [Fig. 11(a)]. As expected, C-slow MCML is more power efficient than multicore CMOS processors at the high-performance regions of the design space.

D. Underutilization

Underutilization of the available hardware threads in an MCML processor can degrade the energy efficiency advantages over CMOS. Fig. 12 compares energy and power between the CMOS and the MCML processors when the number of software threads is fewer than the number of hardware threads. Both the MCML and the CMOS consume more energy and less power when hardware threads are underutilized. The MCML power, however, is largely static and independent of the number of threads, whereas CMOS power scales with fewer threads. The energy of MCML and CMOS processors are similar to one to two software threads running at 2C (CMOS two-core and MCML two-slow single core), and two to four threads at 4C and 8C. The energy and power advantage of MCML processors is maintained, and the maximum energy benefit is obtained when all hardware threads are fully utilized. Hence, systems expected to run workloads with varying degrees of thread-level parallelism would likely benefit from heterogeneous hardware that incorporates both the MCML and the static CMOS cores.

E. Area

An area breakdown of a single-core, a C-slow MCML processor, and a C-core CMOS processor at 0.8 V is shown

in Fig. 11(b). All the memory structures are the same capacity for CMOS and MCML for a given C. A single MCML core, excluding the cache subsystem, consumes 1.89× greater area than a CMOS core due to the differential signaling (wiring area), relatively large MCML gates, and additional MCML–CMOS circuit interfaces. As C increases, the MCML area grows due to the greater number of pipeline registers, interfaces, and extra SRAM resources for more hardware threads. At the same performance level, however, eight static CMOS cores consume 1.47× more area than the eight-slow MCML core, since the area overhead of increasing cores is larger than increasing the pipeline depth. By including the area overhead of larger devices to mitigate the impact of process variability, the eight-slow MCML processor consumes 9% less area than the eight-core CMOS processor.

F. Process Variability

Variability, which includes device die-to-die and within-die process variations (systematic and random), supply voltage fluctuations, and temperature variations inevitably affect the performance and power of both the static CMOS and the MCML circuits. Variations in MCML circuits are primarily due to the imperfections of the bias voltage generator for a constant current, voltage drops in the power and ground networks (typically much less than static CMOS), transistor mismatch between the pMOS load transistors, or the nMOS transistors in the same differential pair, and local temperature differences. MCML circuits can tolerate some variation by adjusting bias voltages. A typical on-chip voltage generator provides the bias voltages to maintain the voltage swing and tail current nearly constant [3].

Four process corners are evaluated to assess the impact of die-to-die variability for both the static CMOS (0.8 V) and the MCML processors. MCML performs worse than CMOS in terms of performance variability, but better than CMOS in terms of power variability due to lower voltage and current fluctuations. Among all corners, the worst performance variation is 24% for CMOS and 43% for MCML, while the worst power variation is 25% for CMOS and 16% for MCML.

To demonstrate robustness under both the random and the systematic within-die variations, 1000 Monte Carlo simulations (Section VII-A) are performed on the eight-slow MCML processor. A histogram of the resultant clock frequency distribution is shown in Fig. 11(c). The mean of the frequency drifts from the nominal 13 to 12.85 GHz, because the critical path differs among the 1000 simulations due to variability. The standard deviation of the critical path frequency is within 4% ($s = 0.473$) of the mean due to averaging effects.

IX. Conclusion

A 22-nm MCML standard cell library has been developed for MCML microprocessors incorporating multithreading and C-slow retiming. A 10–40× reduction in power and ground noise over static CMOS processors at both the low (0.55 V) and the nominal voltages (0.8 V) is observed. An eight-slow (33 stages) single-core MCML microprocessor delivers 1.6× lower energy and 9% less area than an eight-core static CMOS processor, while the performance is within 13%. A dual-core, eight-slow MCML microprocessor outperforms an eight-core CMOS processor by 16% with a 19% improvement in energy. Furthermore, the operation of the MCML processor is robust under both the systematic and the random device variations. Current-mode microprocessors have the potential to significantly improve signal integrity and energy efficiency of future computer systems.

References

[1] J. M. Musicer and J. Rabaey, "MOS current mode logic for low power, low noise CORDIC computation in mixed-signal environments," in *Proc. Int. Symp. Low Power Electron. Design*, 2000, pp. 102–107.

[2] I. Savidis, S. Kose, and E. G. Friedman, "Power noise in TSV-based 3-D integrated circuits," *IEEE J. Solid-State Circuits*, vol. 48, no. 2, pp. 587–597, Feb. 2013.

[3] S. Badel, "MOS current-mode logic standard cells for high-speed low-noise applications," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2008.

[4] *Cray X1E Supercomputer*. [Online]. Available: http://www.cray.com/products/x1e/, accessed 2005.

[5] A. E. Barish, J. P. Eckhardt, M. D. Mayo, W. A. Svarczkopf, and S. P. Gaur, "Improved performance of IBM enterprise system/9000 bipolar logic chips," *IBM J. Res. Develop.*, vol. 36, no. 5, pp. 829–834, Sep. 1992.

[6] E. B. Eichelberger and S. E. Bello, "Differential current switch—High performance at low power," *IBM J. Res. Develop.*, vol. 35, no. 3, pp. 313–320, May 1991.

[7] J. Kolodzey, "CRAY-1 computer technology," *IEEE Trans. Compon., Hybrids, Manuf. Technol.*, vol. 4, no. 2, pp. 181–186, Jun. 1981.

[8] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-way multithreaded SPARC processor," *IEEE Micro*, vol. 25, no. 2, pp. 21–29, Mar./Apr. 2005.

[9] U. G. Nawathe, M. Hassan, K. C. Yen, A. Kumar, A. Ramachandran, and D. Greenhill, "Implementation of an 8-core, 64-thread, power-efficient SPARC server on a chip," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 6–20, Jan. 2008.

[10] S. Phillips, "M7: Next generation SPARC," in *Proc. Hot Chips, Symp. High Perform. Chips*, 2014.

[11] *Intel Xeon Phi Product Family*. [Online]. Available: http://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-detail.html?wapkw=xeon+phi, accessed 2012.

[12] Tilera. *Tilepro Processor Family*. [Online]. Available: http://www.tilera.com/products/processors/TILEPro_Family, accessed 2011.

[13] M. Mizuno *et al.*, "A GHz MOS adaptive pipeline technique using MOS current-mode logic," *IEEE J. Solid-State Circuits*, vol. 31, no. 6, pp. 784–791, Jun. 1996.

[14] J. Cao *et al.*, "OC-192 transmitter and receiver in standard 0.18-μm CMOS," *IEEE J. Solid-State Circuits*, vol. 37, no. 12, pp. 1768–1780, Dec. 2002.

[15] A. Tanabe *et al.*, "0.18-μm CMOS 10-Gb/s multiplexer/demultiplexer ICs using current mode logic with tolerance to threshold voltage fluctuation," *IEEE J. Solid-State Circuits*, vol. 36, no. 6, pp. 988–996, Jun. 2001.

[16] H. T. Bui and Y. Savaria, "Shunt-peaking in MCML gates and its application in the design of a 20 Gb/s half-rate phase detector," in *Proc. ISCAS*, May 2004, pp. IV-369–IV-372.

[17] M. P. Houlgate, D. J. Olszewski, K. Abdelhalim, and L. MacEachern, "Adaptable MOS current mode logic for use in a multi-band RF prescaler," in *Proc. ISCAS*, May 2004, pp. IV-329–IV-332.

[18] M. Alioto and G. Palumbo, "Design strategies for source coupled logic gates," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 50, no. 5, pp. 640–654, May 2003.

[19] O. Musa and M. Shams, "An efficient delay model for MOS current-mode logic automated design and optimization," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 8, pp. 2041–2052, Aug. 2010.

[20] H. Hassan, M. Anis, and M. Elmasry, "MOS current mode circuits: Analysis, design, and variability," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 8, pp. 885–898, Aug. 2005.

[21] A. Cevrero, F. Regazzoni, M. Schwander, S. Badel, P. Ienne, and Y. Leblebici, "Power-gated MOS current mode logic (PG-MCML): A power aware DPA-resistant standard cell library," in *Proc. 48th ACM/EDAC/IEEE Design Autom. Conf.*, Jun. 2011, pp. 1014–1019.

[22] E. Grochowski, D. Ayers, and V. Tiwari, "Microarchitectural dl/dt control," *IEEE Des. Test. Comput.*, vol. 20, no. 3, pp. 40–47, May/Jun. 2003.

[23] W. El-Essawy and D. H. Albonesi, "Mitigating inductive noise in SMT processors," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2004, pp. 332–337.

[24] F. Mohamood, M. B. Healy, S. K. Lim, and H.-H. S. Lee, "A floorplan-aware dynamic inductive noise controller for reliable processor design," in *Proc. MICRO*, Dec. 2006, pp. 3–14.

[25] M. D. Pant, P. Pant, and D. S. Wills, "On-chip decoupling capacitor optimization using architectural level prediction," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 3, pp. 319–326, Jun. 2002.

[26] R. Zhang, K. Skadron, B. H. Meyer, M. R. Stan, and W. Huang, "Some limits of power delivery in the multicore era," in *Proc. WEED*, 2012.

[27] M. S. Gupta, J. L. Oatley, R. Joseph, G.-Y. Wei, and D. M. Brooks, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *Proc. DATE*, Apr. 2007, pp. 1–6.

[28] M. S. Gupta, K. K. Rangan, M. D. Smith, G.-Y. Wei, and D. Brooks, "Towards a software approach to mitigate voltage emergencies," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2007, pp. 123–128.

[29] V. J. Reddi, M. S. Gupta, G. Holloway, G.-Y. Wei, M. D. Smith, and D. Brooks, "Voltage emergency prediction: Using signatures to reduce operating margins," in *Proc. HPCA*, Feb. 2009, pp. 18–29.

[30] M. S. Gupta, K. K. Rangan, M. D. Smith, G.-Y. Wei, and D. Brooks, "DeCoR: A delayed commit and rollback mechanism for handling inductive noise in processors," in *Proc. HPCA*, Feb. 2008, pp. 381–392.

[31] M. S. Gupta, V. J. Reddi, G. Holloway, G.-Y. Wei, and D. M. Brooks, "An event-guided approach to reducing voltage noise in processors," in *Proc. DATE*, Apr. 2009, pp. 160–165.

[32] V. J. Reddi *et al.*, "Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling," in *Proc. 43rd Annu. IEEE/ACM Int. Symp. Architecture (MICRO)*, Dec. 2010, pp. 77–88.

[33] K. T. Tang and E. G. Friedman, "Simultaneous switching noise in on-chip CMOS power distribution networks," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 4, pp. 487–493, Aug. 2002.

[34] M. W. Allam and M. I. Elmasry, "Dynamic current mode logic (DyCML): A new low-power high-performance logic style," *IEEE J. Solid-State Circuits*, vol. 36, no. 3, pp. 550–558, Mar. 2001.

[35] G. Hinton, D. Sager, M. Upton, and D. Boggs, "The microarchitecture of the Pentium 4 processor," *Intel Technol. J.*, 2001.

[36] S. Wijeratne *et al.*, "A 9 GHz 65 nm Intel Pentium 4 processor integer execution core," in *Proc. ISSCC*, Feb. 2006, pp. 353–365.

[37] M. Seok, D. Jeon, C. Chakrabarti, D. Blaauw, and D. Sylvester, "A 0.27 V 30 MHz 17.7 nJ/transform 1024-pt complex FFT core with super-pipelining," in *Proc. ISSCC*, Feb. 2011, pp. 342–344.

[38] C. E. Leiserson, F. M. Rose, and J. B. Saxe, "Optimizing synchronous circuitry by retiming (preliminary version)," in *Proc. 3rd Caltech Conf. Very Large Scale Integr.*, 1983, pp. 87–116.

[39] N. Weaver, Y. Markovskiy, Y. Patel, and J. Wawrzynek, "Post-placement C-slow retiming for the Xilinx Virtex FPGA," in *Proc. ACM/SIGDA 11th Int. Symp. FPGA*, 2003, pp. 185–194.

[40] D. M. Tullsen, S. J. Eggers, and H. M. Levy, "Simultaneous multithreading: Maximizing on-chip parallelism," in *Proc. 22nd ISCA*, Jun. 1995, pp. 392–403.

[41] M. Alioto and G. Palumbo, "Feature—Power-aware design techniques for nanometer MOS current-mode logic gates: A design framework," *IEEE Circuits Syst. Mag.*, vol. 6, no. 4, pp. 42–61, Dec. 2006.

[42] E. Salman and E. G. Friedman, *High Performance Integrated Circuit Design*. New York, NY, USA: McGraw-Hill, 2012.

[43] S. Köse and E. G. Friedman, "Efficient algorithms for fast *IR* drop analysis exploiting locality," *Integr., VLSI J.*, vol. 45, no. 2, pp. 149–161, Mar. 2012.

[44] E. Salman, E. G. Friedman, R. M. Secareanu, and O. L. Hartin, "Worst case power/ground noise estimation using an equivalent transition time for resonance," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 5, pp. 997–1004, May 2009.

[45] K. T. Tang and E. G. Friedman, "Incorporating voltage fluctuations of the power distribution network into the transient analysis of CMOS logic gates," *Analog Integr. Circuits Signal Process.*, vol. 31, no. 3, pp. 249–259, Jun. 2002.

[46] A. V. Mezhiba and E. G. Friedman, *Power Distribution Networks in High Speed Integrated Circuits*. New York, NY, USA: Springer-Verlag, 2012.

[47] J. Zhang and E. G. Friedman, "Crosstalk modeling for coupled RLC interconnects with application to shield insertion," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 6, pp. 641–646, Jun. 2006.

[48] K. T. Tang and E. G. Friedman, "Delay and noise estimation of CMOS logic gates driving coupled resistive–capacitive interconnections," *Integr., VLSI J.*, vol. 29, no. 2, pp. 131–165, Sep. 2000.

[49] V. Vishnyakov, E. G. Friedman, and A. Kolodny, "Multi-aggressor capacitive and inductive coupling noise modeling and mitigation," *Microelectron. J.*, vol. 43, no. 4, pp. 235–243, Apr. 2012.

[50] V. J. Reddi *et al.*, "Voltage noise: Why its bad, and what to do about it," in *Proc. 5th IEEE Workshop Silicon Errors Logic-Syst. Effects (SELSE)*, Palo Alto, CA, USA, Mar. 2009, pp. 1–4.

[51] M. D. Powell and T. N. Vijaykumar, "Pipeline damping: A microarchitectural technique to reduce inductive noise in supply voltage," *ACM SIGARCH Comput. Archit. News*, vol. 31, no. 2, pp. 72–83, May 2003.

[52] V. J. Reddi *et al.*, "Voltage noise in production processors," *IEEE Micro*, vol. 31, no. 1, pp. 20–28, Jan./Feb. 2011.

[53] M. D. Pant, P. Pant, D. S. Wills, and V. Tiwari, "Inductive noise reduction at the architectural level," in *Proc. 13th Int. Conf. VLSI Design*, Jan. 2000, pp. 162–167.

[54] J. Musicer, "An analysis of MOS current mode logic for low power and high performance digital logic," Ph.D. dissertation, Citeseer, 2000.

[55] P. R. Gray, R. G. Meyer, P. J. Hurst, and S. H. Lewis, *Analysis and Design of Analog Integrated Circuits*. New York, NY, USA: Wiley, 2001.

[56] V. Srinivasan *et al.*, "Optimizing pipelines for power and performance," in *Proc. MICRO*, 2002, pp. 333–344.

[57] A. Hartstein and T. R. Puzak, "Optimum power/performance pipeline depth," in *Proc. MICRO*, Dec. 2003, pp. 117–125.

[58] *Floating Point Unit*. [Online]. Available: http://opencores.org/project/fpu, accessed 2014.

[59] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm design exploration," in *Proc. Int. Symp. Quality Electron. Design*, Mar. 2006, pp. 584–590.

[60] W. Snyder, "Verilator and systemperl," in *Proc. North Amer. SystemC Users' Group, Design Autom. Conf.*, 2004.

[61] *Incisive Enterprise Simulator*. [Online]. Available: http://www.cadence.com/products/fv/enterprise_simulator, accessed 2014.

[62] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0," in *Proc. MICRO*, 2007, pp. 3–14.

[63] P. M. Ortego and P. Sack, "SESC: SuperESCalar simulator," in *Proc. 17th Euro Micro Conf. Real Time Syst. (ECRTS)*, 2004, pp. 1–4.

[64] *The FreePDKTM Process Design Kit*. [Online]. Available: http://www.eda.ncsu.edu/wiki/FreePDK, accessed 2011.

[65] H. Hassan, M. Anis, and M. Elmasry, "Design and optimization of MOS current mode logic for parameter variations," *Integr., VLSI J.*, vol. 38, no. 3, pp. 417–437, Jan. 2005.

[66] X. Liang and D. Brooks, "Mitigating the impact of process variations on processor register files and execution units," in *Proc. MICRO*, Dec. 2006, pp. 504–514.

[67] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "VARIUS: A model of process variation and resulting timing errors for microarchitects," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 1, pp. 3–13, Feb. 2008.

[68] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, "Matching properties of MOS transistors," *IEEE J. Solid-State Circuits*, vol. 24, no. 5, pp. 1433–1439, Oct. 1989.

[69] Synopsys Inc. *Design Compiler User Guide*. [Online]. Available: http://www.synopsys.com, accessed 2011.

[70] L. Xiao *et al.*, "Local clock skew minimization using blockage-aware mixed tree-mesh clock network," in *Proc. ICCAD*, Nov. 2010, pp. 458–462.

[71] R. Jakushokas, M. Popovich, A. V. Mezhiba, S. Köse, and E. Friedman, *Power Distribution Networks With On-Chip Decoupling Capacitors*. New York, NY, USA: Springer-Verlag, 2010.

[72] H. Bhatnagar, *Advanced ASIC Chip Synthesis: Using Synopsys Design Compiler Physical Compiler and PrimeTime*. New York, NY, USA: Springer-Verlag, 2007.

[73] S. Badel *et al.*, "A generic standard cell design methodology for differential circuit styles," in *Proc. DATE*, Mar. 2008, pp. 843–848.

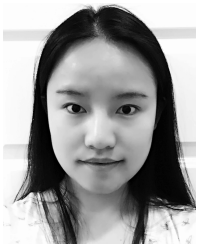
[74] K. Tiri and I. Verbauwhede, "Place and route for secure standard cell design," in *Proc. CARDIS*, 2004, pp. 143–158.

[75] H. H. Chen and D. D. Ling, "Power supply noise analysis methodology for deep-submicron VLSI chip design," in *Proc. 34th Design Autom.*, Jun. 1997, pp. 638–643.

[76] A. Devgan and S. Nassif, "Power variability and its impact on design," in *Proc. Int. VLSI Design*, Jan. 2005, pp. 679–682.

[77] R. Chaudhry, R. Panda, T. Edwards, and D. Blaauw, "Design and analysis of power distribution networks with accurate RLC models," in *Proc. Int. VLSI Design*, Jan. 2000, pp. 151–155.

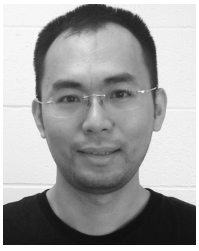
[78] A. Dharchoudhury, R. Panda, D. Blaauw, R. Vaidyanathan, B. Tutuianu, and D. Bearden, "Design and analysis of power distribution networks in PowerPC microprocessors," in *Proc. Design Autom.*, Jun. 1998, pp. 738–743.
[79] Micron. (Mar. 2006). *512Mb DDR2 SDRAM Component Data Sheet: MT47H128M4B6-25*. [Online]. Available: http://www.micron.com/~/media/Documents/Products/Data%20Sheet/DRAM/1Gb_DDR2.pdf
[80] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory access scheduling," in *Proc. 27th ISCA*, 2000, pp. 128–138.
[81] R. M. Yoo, A. Romano, and C. Kozyrakis, "Phoenix rebirth: Scalable MapReduce on a large-scale shared-memory system," in *Proc. IEEE Int. Symp. Workload Characterization*, Oct. 2009, pp. 198–207.
[82] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," in *Proc. 22nd ISCA*, Jun. 1995, pp. 24–36.
[83] L. Dagum and R. Menon, "OpenMP: An industry standard API for shared-memory programming," *IEEE Comput. Sci. Eng.*, vol. 5, no. 1, pp. 46–55, Mar. 1998.
[84] D. H. Bailey *et al.*, "The NAS parallel benchmarks," *Int. J. High Perform. Comput. Appl.*, vol. 5, no. 3, pp. 63–73, 1991.
[85] R. D. Lawrence, G. S. Almasi, and H. E. Rushmeier, "A scalable parallel algorithm for self-organizing maps with applications to sparse data mining problems," *Data Mining Knowl. Discovery*, vol. 3, no. 2, pp. 171–195, 1999.



Yuxin Bai received the B.S. degree in electrical and computer engineering from the Harbin Institute of Technology, Harbin, China, in 2010, and the M.S. degree in electrical and computer engineering from the University of Rochester, Rochester, NY, USA, in 2012, where she is currently pursuing the Ph.D. degree in electrical and computer engineering.

Her current research interests include energy-efficient microarchitectures, MOS current-mode logic, and intelligent power management techniques.

Ms. Bai was a recipient of the Freescale Semiconductor Fellowship in 2008 and 2009.



Yanwei Song received the B.S. degree in instrument science and technology from Zhejiang University, Hangzhou, China, in 2005, and the M.S. degree in electrical and computer engineering from the University of Rochester, Rochester, NY, USA, in 2010, where he is currently pursing the Ph.D. degree.

His current research interests include energy-efficient architecture, including both processor and memory systems.

Mahdi Nazm Bojnordi received the M.S. degree in electrical and computer engineering from the University of Tehran, Tehran, Iran. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Rochester, Rochester, NY, USA.

His current research interests include designing main memory controllers capable of performing application-specific performance and energy-efficiency optimizations.

Mr. Bojnordi was a recipient of the 2013 IEEE Micro Top Picks Award.

Alexander Shapiro received the B.S. degree in computer engineering from the Technion–Israel Institute of Technology, Haifa, Israel, in 2010, and the M.S. degree in electrical engineering from the University of Rochester, Rochester, NY, USA, in 2012, where he is currently pursuing the Ph.D. degree in electrical engineering.

He held a variety of hardware research and development positions with IBM, Petah Tikva, Israel, and Intel, Petah Tikva, from 2008 to 2011. His current research interests include the analysis and design of high performance integrated circuits, low power techniques, and near threshold circuits.

Eby G. Friedman (F'00) received the B.S. degree from Lafayette College, Easton, PA, USA, in 1979, and the M.S. and Ph.D. degrees from the University of California at Irvine, Irvine, CA, USA, in 1981 and 1989, respectively, all in electrical engineering.

He was with Hughes Aircraft Company, Glendale, CA, USA, from 1979 to 1991, as the Manager of the Department of Signal Processing Design and Test, responsible for the design and test of high performance digital and analog ICs. He has been with the Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY, USA, since 1991, where he is currently a Distinguished Professor, and the Director of the High Performance VLSI/IC Design and Analysis Laboratory. He is also a Visiting Professor with the Technion–Israel Institute of Technology, Haifa, Israel. He has authored over 400 papers and book chapters, 12 patents, and authored and edited 16 books in high speed and low power CMOS design techniques, 3-D design methodologies, high speed interconnect, and the theory and application of synchronous clock and power distribution networks. His current research interests include high performance synchronous digital and mixed-signal microelectronic design and analysis with application to high-speed portable processors and low power wireless communications.

Dr. Friedman is a Senior Fulbright Fellow. He was a recipient of the IEEE Circuits and Systems Charles A. Desoer Technical Achievement Award in 2013, the University of Rochester Graduate Teaching Award, and the College of Engineering Teaching Excellence Award. He is the Editor-in-Chief of the *Microelectronics Journal*, a member of the Editorial Boards of *Analog Integrated Circuits and Signal Processing*, the *Journal of Low Power Electronics*, and the *Journal of Low Power Electronics and Applications*, the Chair of the IEEE TRANSACTIONS ON VLSI SYSTEMS Steering Committee, and a member of the Technical Program Committee of numerous conferences. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON VLSI SYSTEMS, the Regional Editor of the *Journal of Circuits, Systems and Computers*, a member of the Editorial Board of the PROCEEDINGS OF THE IEEE, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: ANALOG AND DIGITAL SIGNAL PROCESSING, the IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS, and the *Journal of Signal Processing Systems*, a member of the Circuits and Systems Society Board of Governors, and the Technical Program Chair of several IEEE conferences.

Engin Ipek received the B.S., M.S., and Ph.D. degrees from Cornell University, Ithaca, NY, USA, in 2003, 2007, and 2008, respectively, all in electrical and computer engineering.

He was a Researcher with the Computer Architecture Group, Microsoft Research, from 2007 to 2009. He is currently an Associate Professor of Electrical and Computer Engineering and Computer Science with the University of Rochester, Rochester, NY, USA, where he leads the Computer Systems Architecture Laboratory. His current research interests include energy-efficient architectures, high performance memory systems, and the application of emerging memory technologies to computer systems.

Prof. Ipek's research has been recognized by the 2014 IEEE Computer Society TCCA Young Computer Architect Award, two IEEE Micro Top Picks Awards, an Invited Communications of the ACM research highlights article, an ASPLOS 2010 Best Paper Award, and an NSF CAREER Award.