KiloCore: A 32-nm 1000-Processor Computational Array

Brent Bohnenstiehl, *Student Member, IEEE*, Aaron Stillmaker, *Member, IEEE*, Jon J. Pimentel, *Student Member, IEEE*, Timothy Andreas, *Student Member, IEEE*, Bin Liu, *Student Member, IEEE*, Anh T. Tran, *Member, IEEE*, Emmanuel Adeagbo, *Student Member, IEEE*, and Bevan M. Baas, *Senior Member, IEEE*

***Abstract*— A processor array containing 1000 independent processors and 12 memory modules was fabricated in 32-nm partially depleted silicon on insulator CMOS. The programmable processors occupy 0.055 mm^2 each, contain no algorithm-specific hardware, and operate up to an average maximum clock frequency of 1.78 GHz at 1.1 V. At 0.9 V, processors operating at an average of 1.24 GHz dissipate 17 mW while issuing one instruction per cycle. At 0.56 V, processors operating at an average of 115 MHz dissipate 0.61 mW while issuing one instruction per cycle, resulting in an energy consumption of 5.3 pJ/instruction. On-die communication is performed by complementary circuit and packet-based networks that yield a total array bisection bandwidth of 4.2 Tb/s. Independent memory modules handle data and instructions and operate up to an average maximum clock frequency of 1.77 GHz at 1.1 V. All processors, their packet routers, and the memory modules contain unconstrained clock oscillators within independent clock domains that adapt to large supply voltage noise. Compared with a variety of Intel i7s and Nvidia GPUs, the KiloCore at 1.1 V has geometric mean improvements of 4.3× higher throughput per area and 9.4× higher energy efficiency for AES encryption, 4095-b low-density parity-check decoding, 4096-point complex fast Fourier transform, and 100-B record sorting applications.**

***Index Terms*— Globally asynchronous locally synchronous (GALS), many core, multicore, NoC, parallel processor.**

I. INTRODUCTION

IMPORTANT computing applications of the future range from embedded Internet-of-Things devices to cloud datacenters and are characterized by an increased emphasis on high energy efficiency in addition to high performance [1].

Semiconductor fabrication technologies have fortunately continued to provide increasing levels of integration [2] and provide interesting possibilities for new architectural designs with potential usages both as standalone systems and as components in heterogeneous systems [3]. The performance and efficiency gains possible through parallel processing [4] are well known [5] and substantial effort has been made to integrate many processors onto a single die rather than increase the complexity of a smaller number of processors [6]–[9]. As well as increasing capabilities, the ever increasing costs of fabrication have motivated the search for programmable and/or reconfigurable processors that are not tailored to a single application or a small class of applications and may be scaled to address widely varying computing domains. The presented chip addresses all of the aforementioned factors in a massively parallel computing platform that is easily scalable, energy efficient under a wide variety of conditions, capable of very high performance, and suitable for a broad range of applications or critical kernels from embedded to the cloud as a standalone engine or as a coprocessor in a heterogeneous system.

Section II provides an overview of the chip's architecture. Section III describes the clocking methodologies and circuits used throughout the system that provide high efficiencies and robust operation. Section IV covers the design and CMOS implementation of the chip. Section V presents measured results. Section VI provides application results and comparisons, and Section VII concludes this paper.

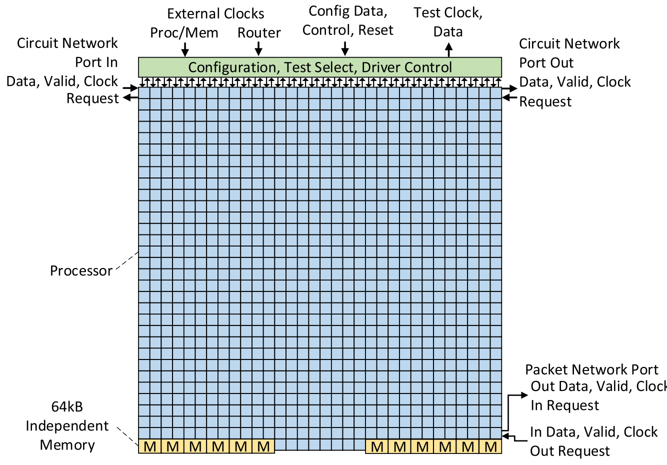

External Clocks
Proc/Mem
Router
Config Data, Control, Reset
Test Clock, Data
Circuit Network Port In Data, Valid, Clock
Request
Configuration, Test Select, Driver Control
Circuit Network Port Out Data, Valid, Clock
Request
Processor
Packet Network Port Out Data, Valid, Clock In Request
64kB Independent Memory
In Data, Valid, Clock Out Request
M

Fig. 1. Top-level processor array diagram.

II. HIGH-LEVEL ARCHITECTURE

The KiloCore chip includes 1000 independent, uniform, programmable, RISC-type, in-order, single-issue processors; and 12 independent memory modules [10]. Processors are arrayed in 32 columns and 31 rows with eight processors and 12 independent memories in a 32nd row, as shown in Fig. 1.

Manuscript received August 8, 2016; revised October 17, 2016; accepted November 26, 2016. Date of publication February 14, 2017; date of current version March 23, 2017. This paper was approved by Guest Editor Makoto Ikeda.

B. Bohnenstiehl, J. J. Pimentel, T. Andreas, B. Liu, E. Adeagbo, and B. M. Baas are with the University of California, Davis, CA 95616 USA (e-mail: bvbohnen@ucdavis.edu; bbaas@ucdavis.edu).

A. Stillmaker is now with California State University, Fresno, CA 93740 USA.

A. T. Tran is now with Cavium Inc., San Jose, CA 95131 USA.

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

Digital Object Identifier 10.1109/JSSC.2016.2638459

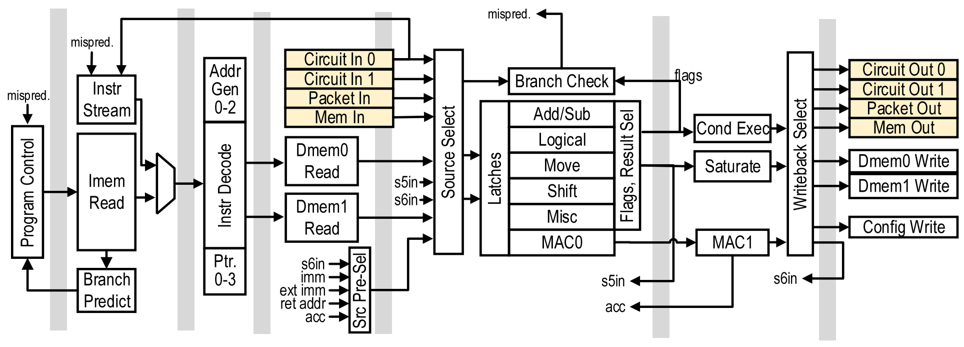


Fig. 2. Major components and connections of the seven-stage processor pipeline. Many control and configuration signals are omitted for clarity.

Processors and independent memory modules with no work to do dissipate exactly zero active power (leakage only)—this is an important capability in the 1000-processor-chip era due to the difficulty in implementing complex software workloads that spread evenly over thousands of processors, which leads to the increasing prevalence of processors with widely varying activity levels [11]. Under most conditions, the processor array has a near-optimal proportional scaling of power dissipation over a wide range of activity levels.

A. Processors

Each processor contains a 128 $\times$ 40-b instruction memory, 512 B of data memory, three programmable data address generators, two 32 $\times$ 16-b input buffers, and a 16-b fixed-point datapath with a 32-b multiplier output and a 40-b accumulator. The 72 instruction types include signed and unsigned operations to enable efficient scaling to 32 b or larger word widths, with no instructions being algorithm-specific. Processors support predication for any instruction using two conditional execution masks, static branch prediction, and automated hardware looping for accelerating inner loops. Although the natural word width of the datapaths and memories is 16-b, through software other word widths are easily handled—for example, 32-b floating point [12] and 10-B sorting keys for 100-B data records [13].

Each processor issues one 40-b instruction in-order per cycle into its seven-stage pipeline (shown in Fig. 2) from its local instruction memory, and it may also source large programs from an on-die independent memory module. Instruction input operands and output results commonly come from or go to the local data memory, one of several circuit-switched network ports, the packet router port, an attached independent memory, a pipeline forwarding path, or a series of special dedicated-purpose registers that include dereferenceable pointers, address generator configuration, predication flag masks, oscillator frequency selection, and other software-accessible core configuration fields.

B. On-Die Communication

The processor array connects processors and independent memories via a 2-D mesh, a topology which maps well to planar integrated circuits and scales simply as the number of processors per die increases. Communication on-chip is accomplished by two complementary means: a very high-throughput and low-latency circuit-switched network [14] and a very-small-area packet router [15]; details are provided in Fig. 3.

The circuit-switched links are source-synchronous, so the source clock travels with the data to the destination, where it is translated to the destination-processor's clock domain. The network supports communication between adjacent and distant processors, as resources allow, with each link supporting a maximum rate of 28.5 Gb/s with optionally inserted registers to maintain data integrity over long distances. Each of the four edges of each processor has two such links entering and two links exiting the processor. The high-throughput circuit-switched network is especially efficient—transferring data to an adjacent processor dissipates 59% less energy than writing and later reading that data using local data memory, and transferring that data to a processor four tiles away requires only 1% more energy than using local data memory.

The packet router inside each processor occupies only 9% of each processor's area and is especially effective for high fan-in and high fan-out communication, as well as for administrative messaging. Each router supports 45.5 Gb/s of throughput with a maximum of 9.1 Gb/s per port. Routers operate autonomously from their host processors and contain their own clock oscillators, so they can power down to zero active power when there are no packets to process. Each router contains five 4 $\times$ 18-b input buffers, one for each cardinal direction and one for the local processor. Routers utilize wormhole routing to efficiently transfer long data bursts, in which a header packet will reserve a path and is followed by an arbitrary number of data packets, terminating in a tail packet which releases the path.

Each circuit or packet link terminates in a dual-clock FIFO memory [16], which reliably transfers data between clock domains. In addition, links contain the necessary asynchronous wake-up signals, which inform idle modules when they need to activate their local clock to process new work or to verify when FIFOs are full or empty. Both network types contribute to a total bisection bandwidth of 4.2 Tb/s.

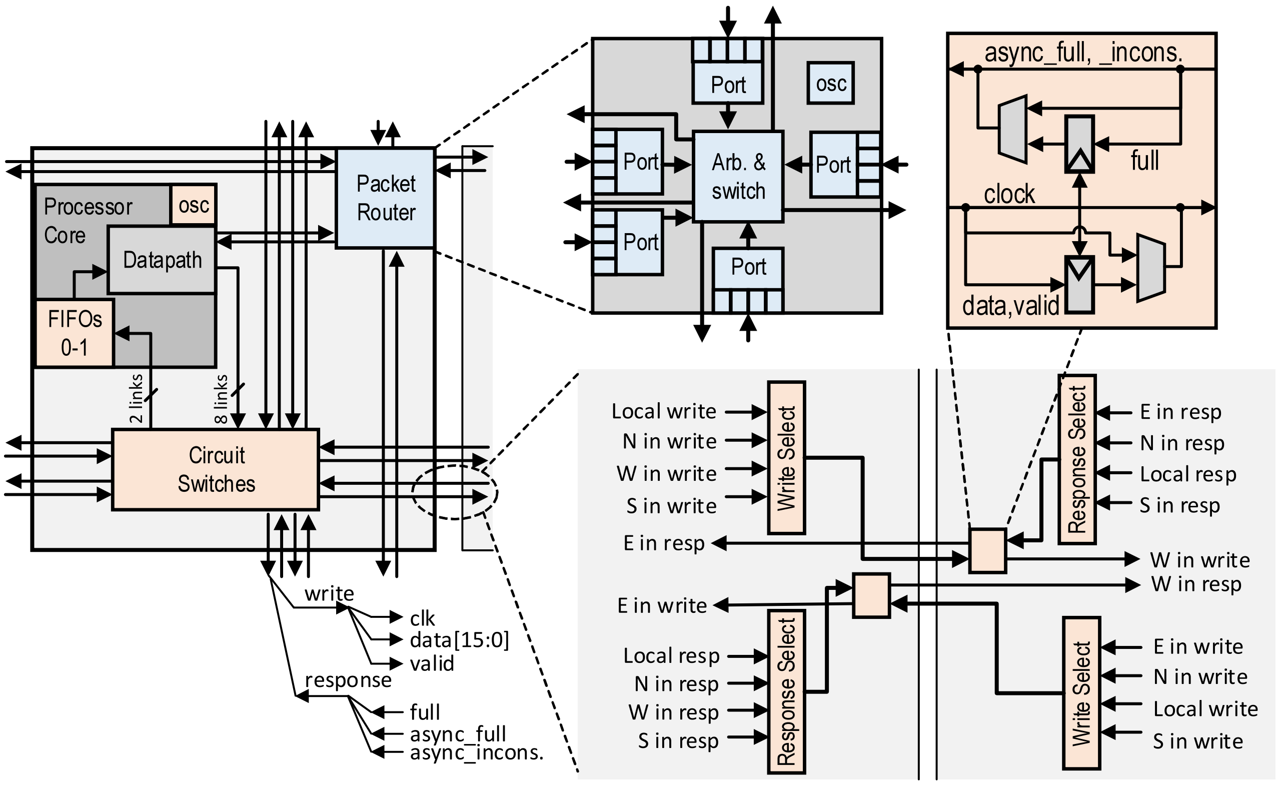


Fig. 3. Overview of intercore communication using circuit and packet networks. Writes are source-synchronous; responses include asynchronous wake-up signals for sleeping processors. Circuit links include configurable registers and an east–west connection for one layer is expanded on the right.

C. Processor Data Memory Organization

Processors with a relatively small amount of memory per core require that memory is used efficiently. A straightforward solution to sustain a throughput of one instruction per cycle with common two-input-operand and one-output-operand instructions would be to utilize an N-word three-port data memory, which is unfortunately not very area or power efficient. If a three-port memory is unavailable, one can be made easily albeit very inefficiently, from two N-word memories with write ports shorted together and reads made independently [7], [17]. A third possibility is to utilize two independent N-word memories, which has the great advantage of yielding a total data space of $2N$ words and being able to sustain two reads and one write per cycle, but only if there are no conflicts where both input operands are in one of the two banks. Conflicts can be resolved by detecting their occurrence and stalling the processor when they occur. We have chosen a hybrid approach, which uses compile-time information to place data into banks to minimize conflicts. When conflicts cannot be avoided or ruled out, data are written into both banks, eliminating the conflict for that datum and allowing a sustained throughput of one instruction per cycle at a cost of the loss of one otherwise-useful data word. Profiles of five diverse applications (AES encryption, 4095-b code length low-density parity-check (LDPC) decoder, 100-B database record sorting, 802.11a/g OFDM Wi-Fi receiver, and software single-precision floating-point arithmetic) showed that 99.66% of all operands across all applications could be mapped to an address in only one bank, and thus, only a very small number of operands needed to be written to both banks redundantly to avoid conflicts during subsequent reads. The scheme permits conflict-free addressing with optimal memory space maximization. Fig. 4 shows the circuitry to implement the three types of writes (bank0, bank1, and both banks) and properly route read data to the processor.

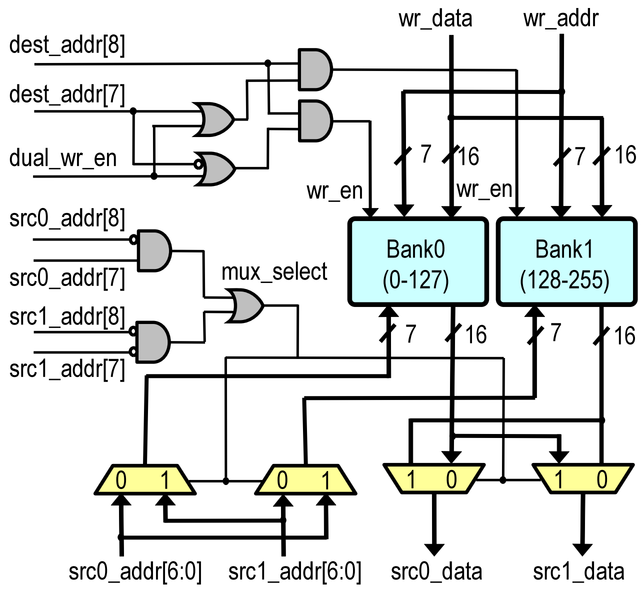


Fig. 4. Multibank data memory read and write circuitry.

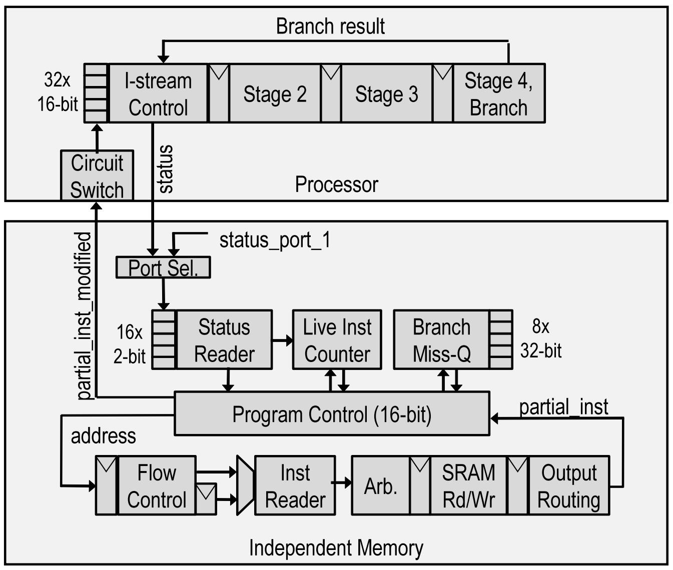


Fig. 5. Components used in streaming instructions from a shared memory to a neighboring processor. Streaming logic is shared between two processors, with only the port 0 connection shown here.

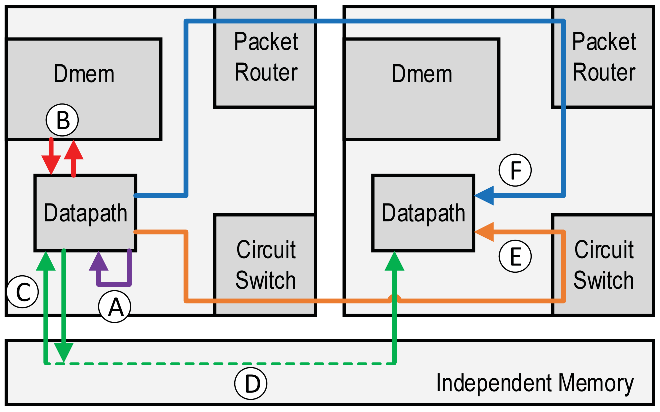
D. Independent Memory Modules

Independent memory modules each contain a 64-kB SRAM and are shared between two neighboring processors. Modules support random and a variety of programmable burst access patterns for data reading and writing, and are also capable of streaming instructions for large-program execution to an adjoining processor using an internal control module. When executing an instruction stream from an independent memory, a processor transfers program control and branch prediction control to dedicated circuits inside the memory block to more efficiently execute across branches. Each memory module contains two 32 × 18-b input buffers, two 32 × 16-b output buffers, and one 16 × 2-b processor response buffer, and supports 28.4 Gb/s of I/O bandwidth. Fig. 5 gives the details of the module's internal blocks.

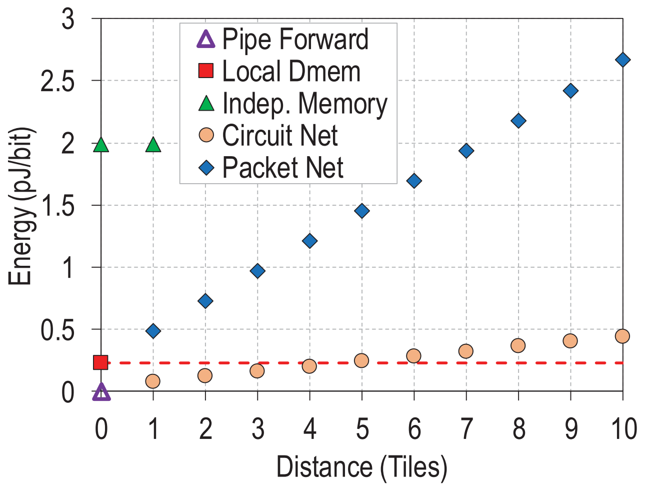
Fig. 6(a) shows the various methods of transferring data from one point in an application to another. These points may be within a single processor or spread across different processors depending on how code has been partitioned. Fig. 6(b) reports the energy costs for each method and includes both a write and a single read, implying transferred data are used only once. Since pipeline forwarding is the lowest-energy method, energy values are reported as additional energy required beyond forwarding, that is, pipeline forwarding = 0.0 in this graph.

III. Fine-Grain Clocking

Many-core applications often require processors to remain idle or operate at low activity for substantial periods of time. Therefore, energy-efficient many-core designs must adapt to wide variations in core workloads. In KiloCore, each core, each packet router inside each core, and each independent memory module contains its own local programmable clock oscillator in an independent fully synchronous clock domain [18], resulting in a total of 2012 globally asynchronous locally synchronous (GALS) [19] clock domains.



(a)



(b)

Fig. 6. Path diagram (a) and measured energies (b) to transfer a bit of data from one point in an application to another versus distance, in addition to the energy required for pipeline forwarding (i.e., pipeline forwarding = 0.0). (A) Pipeline forwarding or (B) local Dmem may be used for in-core transfers. Independent memory may be used for (C) local or (D) neighbor-processor transfers. Both (E) circuit and (F) packet networks support distant transfers.

Oscillators do not use PLLs and each one is allowed to change its frequency, halt, or restart arbitrarily including with respect to other clock domains. Halting is very helpful in saving energy when there is no work to do, which is detected by the processor when it attempts to read an empty input buffer or when attempting to write a full output buffer. Oscillator halting is handled automatically by local hardware logic, which observes instruction source and destination operands, and also the state of interprocessor buffers for both upstream inputs and downstream outputs. When an oscillator is halted, the core/router/memory consumes zero active power. A halted processor consumes only 1.1% of its typical active power through leakage. Oscillators restart in response to asynchronous signals from connected cores when they send data to an empty buffer or free room in a full buffer, for upstream and downstream links, respectively. Cores exiting a halt state require up to three cycles to read input buffers before program execution may continue; cores entering a halt state require a variable number of cycles after program execution pauses to complete any pending writes to the communication network. This inefficiency is negligible

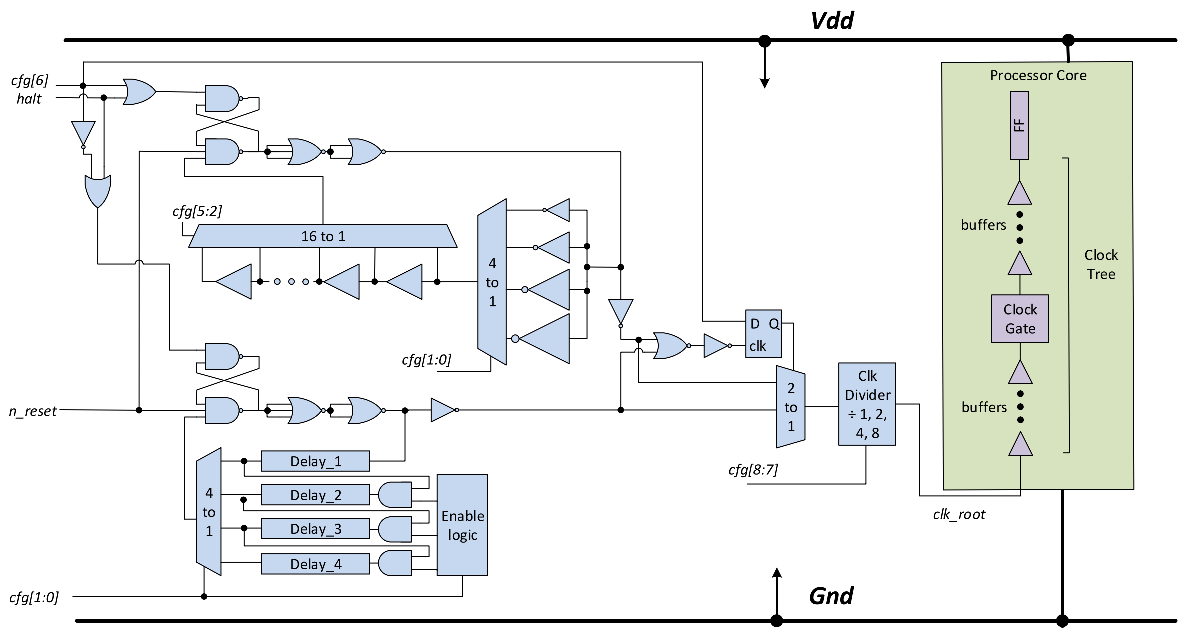


Fig. 7. Digitally programmable clock oscillators, halting and configuration logic, and clocking system used in processors and independent memories.

in many cases; however, it can be significant in situations where high-workload and low-workload cores are connected and performing fine-grain communication, such that the low-workload core is regularly waiting on the high-workload core, but not for long enough periods of time to benefit from clock halting. Per-core oscillator frequency tuning is used to help in this situation, slowing the low-workload core, such that its data production or consumption rate is matched to the high-workload core. We estimate the ideal clock frequency for each core to be the lowest frequency at which a core may operate without reducing overall application throughput. These frequencies are identified during application profiling. Fortunately, even significantly inaccurate frequency estimates typically result in small increases in power dissipation over the ideal case. Tuning could certainly also be done during a run-time tuning phase or even during program execution by a dedicated hardware controller [7]. For the four applications described in Section VI, on average, clock halting yields a 61% reduction in energy usage compared with processors which never halt and do not utilize per-core frequency tuning. Of the energy that is consumed, 87% is from program computation and leakage while 13% is from stall cycles.

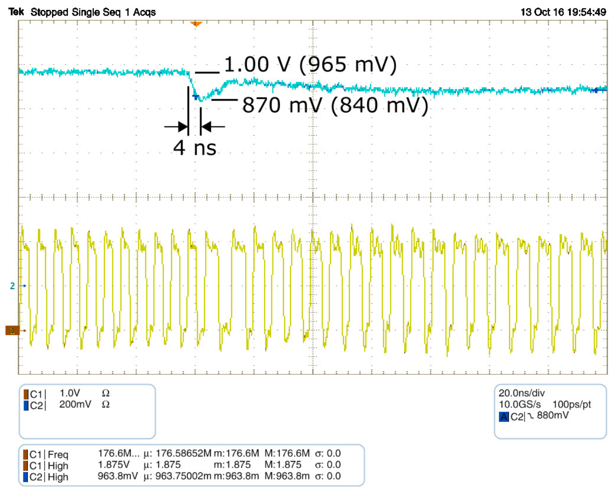
The clock oscillators inside processors and independent memories are composed of two separate ring oscillators, as shown in Fig. 7. The lower oscillator is used to generate low frequencies. The main oscillator utilizes six configuration bits for 64 frequency selections and the low-frequency oscillator utilizes two bits for four selections. Both oscillators divide their output by 1, 2, 4, or 8 before the root clock enters the main clock tree. Each packet router contains a low-area oscillator with four frequency selections.

A. Tolerating Power Grid Voltage Variations

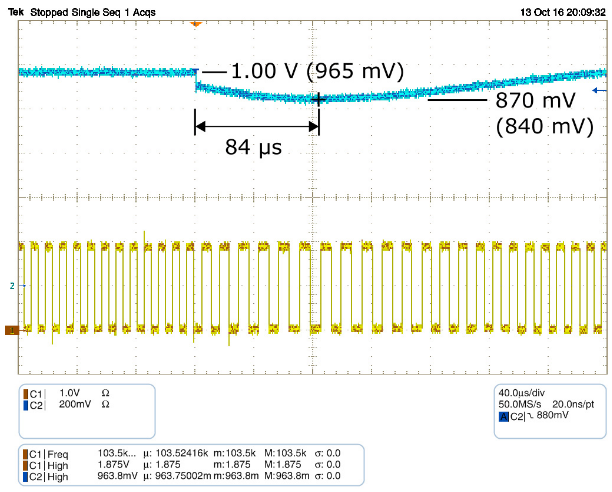
One thousand cores arbitrarily switching between being halted with leakage only to fully active can clearly result in significant power grid noise. Rather than trying to minimize the noise, clock oscillators are designed to rapidly adjust their instantaneous frequencies to compensate for supply noise variations through circuit design and by being powered by the local core's power grid, as shown in Fig. 7. Oscillators are designed so that their frequency tracks closely below the core's maximum operating frequency when voltage droop occurs, and in fact, cores were found to operate error-free when configured to operate at their maximum frequency without any additional margin for voltage droop, though some margin may be needed for overshoot depending on the power supply characteristics. In a manner similar to oscillators, circuit elements in the clock trees, such as buffers and clock gates, naturally adjust their instantaneously delay, because they too are powered by the core's local power grid.

Fig. 8 shows measured waveforms from a beyond-worst case voltage droop event, where 999 processors are simultaneously turned on. In actual usage, only approximately 2/3 of the array could start at one time, because halted cores are restarted by the arrival of external data sent from nonhalted cores, and processors with outputs connected to more than two other processors at a time are very rare in our explored applications. In this test, a single victim processor in the center of the array at coordinates (15,15) runs a critical path test program while the other 999 cores in the array are simultaneously turned on to maximum frequency and begin running an energy intensive program. A globally broadcast configuration signal is used to synchronize this event. The test is performed at a nominal supply voltage of 1.0 V. The victim processor was found to operate error-free throughout the event at a nominal clock frequency equal to its measured standalone maximum frequency to within 20 MHz, the oscillator step size at the test voltage.

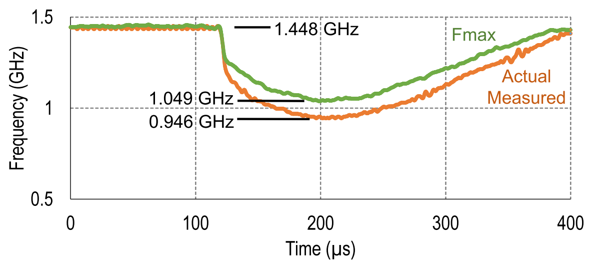
A measured on-die waveform of the supply voltage is shown in Fig. 8(a), showing the short term supply noise using a



(a)



(b)



(c)

Fig. 8. Supply voltage noise at a nominal 1.0 V when simultaneously turning on 999 processors from fully halted to fully active at maximum frequency, while measuring the clock oscillator of a single victim core in the center of the array; 1.0 V appears as 965 mV and 870 mV appears as 840 mV due to a resistor divider created by our 1.9-Ω SMA cable and 50-Ω scope input. (a) 20-ns/division waveform capture, showing a clock frequency reduction in response to a 14% supply voltage reduction over 4 ns. (b) 40-μs/division waveform capture, showing gradual supply droop and recovery, with a corresponding clock frequency recovery. (c) (Lower orange waveform) Instantaneous clock frequency calculated from the time-domain waveform in (b), and (upper green waveform) estimated Fmax derived from independent measurements, showing victim processor operation below but a maximum of 10% from its maximum possible operating frequency Fmax.

200-ns capture window. A 13% reduction in voltage occurs over 4-ns, with a corresponding decrease in the victim processor's clock frequency to compensate. Fig. 8(b) shows the long term voltage droop and recovery using a 400-μs capture window; 84 μs after the turn-on event, the voltage begins recovering from 13% below nominal as a function of the PCB and bench power supply electrical environment. To maintain visibility of the clock waveforms, the processor output clock is divided by 8 for Fig. 8(a) and is reduced to a 630-MHz source frequency and divided by 8192 for Fig. 8(b). Fig. 8(c) shows the victim processor's instantaneous frequency (labeled "Actual Measured") during the test using a higher clock rate data capture, and with the same timescale as Fig. 8(b). Also plotted is the processor's maximum supported frequency corresponding to the instantaneous voltage (labeled "Fmax"), based on pretest measurements. The victim processor's oscillator remains below but within 10% of this maximum Fmax frequency; 84 μs after the turn-on event, the maximum frequency is reduced to 28% below nominal, while the oscillator's frequency is reduced 35%.

IV. Design and Implementation

The processor array is built from standard cells and was synthesized except for small circuits, such as the clock oscillator, which were designed by hand. Cell placement and routing were performed by industry-standard CAD tools. Except for the 64-kB SRAMs inside the independent memory modules, all memories are built from clock-gated flip-flops with synthesized interfacing logic, which greatly simplifies the physical design and lowers the minimum operating voltage for applications which do not use the independent memories.

The 8.0 mm × 8.0 mm chip was fabricated in a 32-nm partially depleted silicon on insulator technology and contains 621 million transistors. The entire array measures 7.94 mm × 7.82 mm. Each processor contains 575 000 transistors and occupies 239 μm × 232 μm; therefore 18 processors occupy almost 1 mm^2. Fig. 9 is a die micrograph showing outlines of the 1000 cores and 12 independent memories, and 564 C4 solder bumps for flip-chip mounting in the center of the array. The chip is mounted inside a stock 676-ball BGA package that delivers full power to only the approximately 160 central processors; therefore, a maximum execution rate of 1.78 trillion MIMD instructions/s per chip is possible only with a custom-designed chip package. I/O signaling is handled by 64 LVDS drivers and 38 single-ended drivers. Pad drivers are placed along the periphery of the processor array. Ten analog voltage probe points are included to support on-chip voltage measurements.

Fig. 10(a) is a postplacement plot of a single processor tile showing regions of the largest components along with details on the die area occupied by the various components. Both the circuit-switched network (including FIFO0 and FIFO1) and the packet-switched network (includes router clock oscillator) occupy 9% of each tile's area. The processor's two clock oscillators and associated control occupy 1% of the tile's area and recover some of that area by eliminating the need for a chip-level clock tree. Fig. 11 shows the same information for a single independent memory tile.

TABLE I

ENERGY PER OPERATION OR ACTIVITY AT A SUPPLY VOLTAGE OF 900 mV. ROUTER FLIT TRANSFER DOES NOT INCLUDE CLOCK ENERGY; PROCESSOR AND MEMORY OPERATIONS INCLUDE CLOCK ENERGY; BRANCH MISPREDICTION ENERGY INCLUDES THREE HIGH-ACTIVITY INSTRUCTIONS ON THE MISPREDICTED PATH

Operation or Activity	Energy (pJ)
Instruction, ALU Add/Sub	11.0
Instruction, ALU Logic	10.3
Instruction, ALU Move	10.0
Instruction, ALU Shift	9.9
Instruction, ALU Other	9.7
Instruction, MAC	13.1
Instruction, Branch Correct	9.7
Instruction, Branch Incorrect	41.0
No-op	7.5
Stall Cycle	6.9
Dmem Read	1.0
Dmem Write 1 Bank	2.7
Dmem Write 2 Banks	5.0
Circuit Comm. First Tile	1.3
Circuit Comm. Additional Tiles	0.6
Packet Router Clock	2.2
Packet Router Flit Transfer	1.9
Shared Memory Stall Cycle	4.5
Shared Memory Read	12.3
Shared Memory Write	19.6

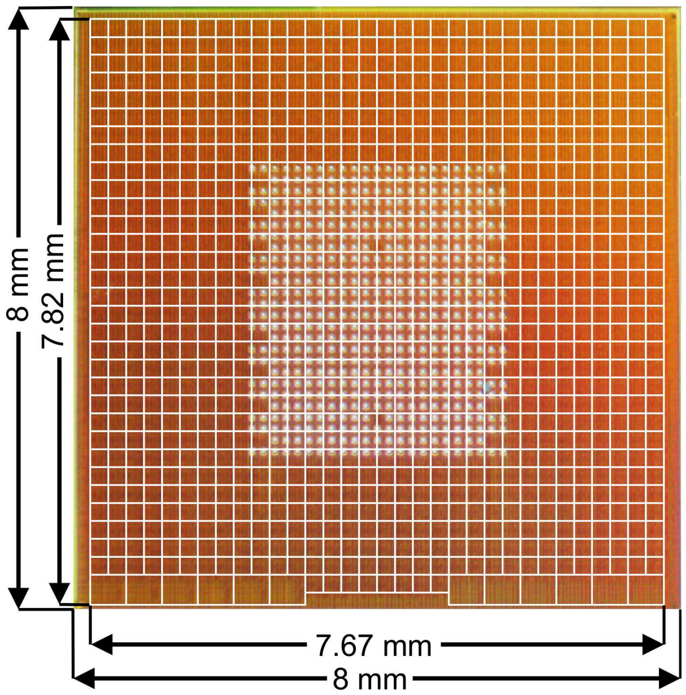
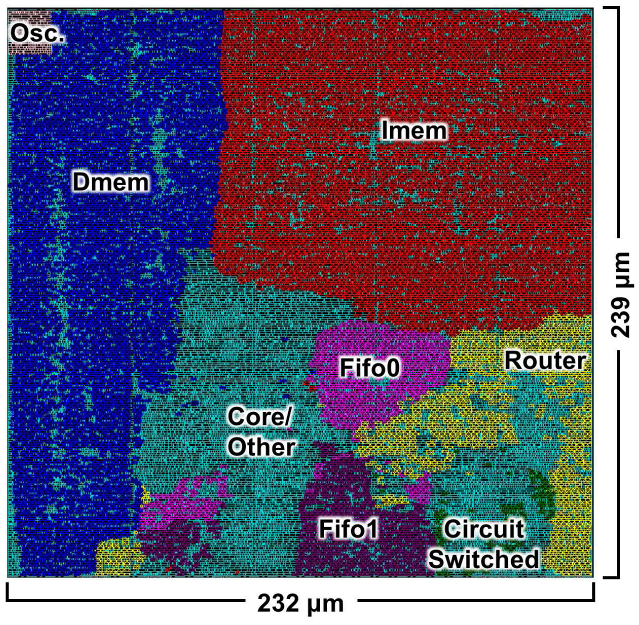


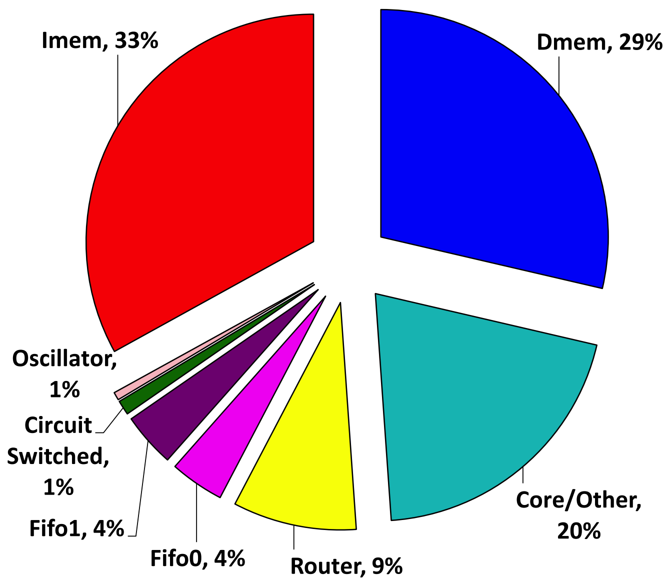
Fig. 9. Die micrograph.

V. MEASURED RESULTS

Processors, routers, and independent memories operate from a maximum voltage of 1.1 V down to minimum voltages of 560, 670, and 760 mV, respectively. Fig. 12 shows the average maximum frequency for each of these modules across their operable ranges. Independent memories have a reduced operating voltage due to their large SRAM array. The reason for the routers' reduced operating voltage is not known but suspected to be due to a specific implementation feature in their GALS network interfaces. Individual cores are allowed to operate at their local Fmax due to GALS clocking. At their highest voltage, processors average 1.78 GHz. When certain critical paths related to ALU carry and zero flags are avoided or coded with two instructions, a processor may operate up to 22% above its normal maximum frequency—a typical processor using this technique was measured operating at 2.29 GHz. This is done by a simple reprogramming of the clock oscillator, so that its frequency is appropriately higher than normal based on the critical paths used by the program assigned to that processor.

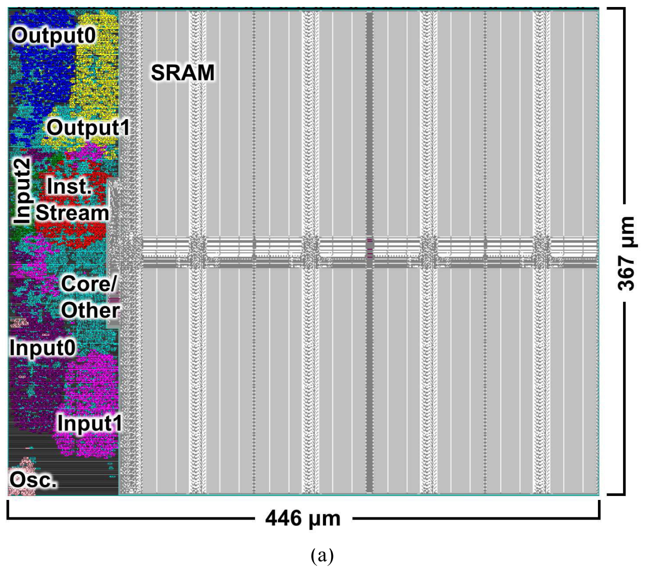


(a)



(b)

Fig. 10. (a) Annotated layout and (b) area breakdown of a single processor tile.



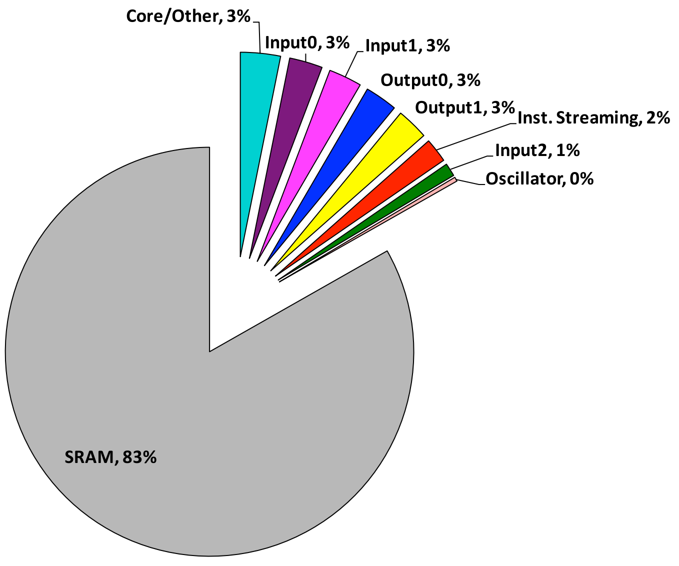


Fig. 11. (a) Annotated layout and (b) area breakdown of a single independent memory tile.

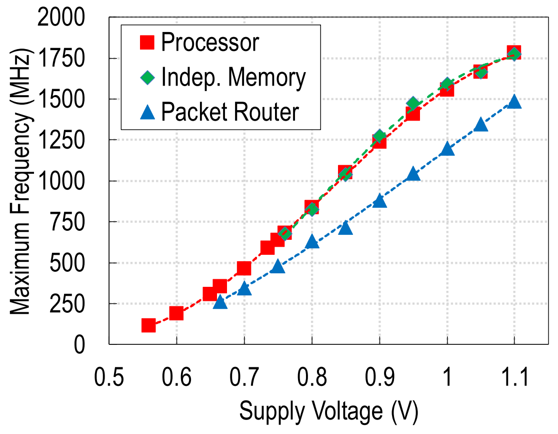


Fig. 12. Maximum operating frequency of processors, memories, and routers.

Table I lists energy usage of a variety of instructions and events when operating at 900 mV. ALU and MAC instructions are categorized according to their pipeline groupings, where input latches for each group isolate them from each other. Measurements are taken with high operand bit activity. A missed timing path through the MAC causes a reduced maximum frequency of operation for MAC instructions, however operation beyond Fmax is possible with a 2-cycle software change.

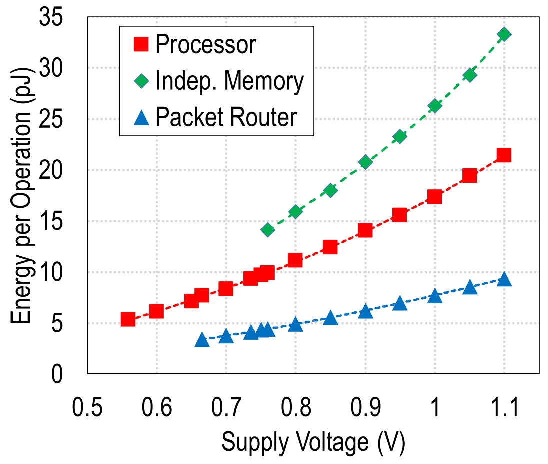


Fig. 13. Energy per typical operation for processors, memories, and routers.

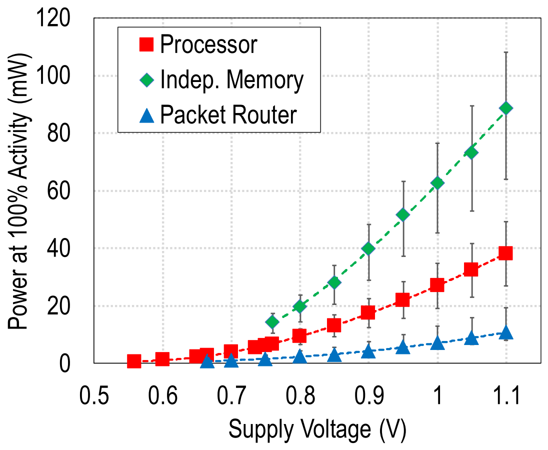


Fig. 14. Power of a processor, memory, and router when 100% active and operating at the maximum clock frequency at the indicated supply voltage.

Fig. 13 shows the typical energy per operation for each module type across its operable voltage range. Processor power varies considerably with instruction selection and memory access patterns. Therefore, processor instruction energy is calculated using weighted averages based on code from a profiled 334-processor fast Fourier transform (FFT) application, including data reads/writes along with circuit network communication. At 560 mV, a single processor dissipates 5.3 pJ per typical instruction while operating at 115 MHz. Packet router energy varies with port activity level; values reported are for transferring a single flit, assuming two router ports are actively transferring and sharing clock energy. Independent memory energy also depends on activity and so values reported are an average of random read and random write energies, and are further averaged between one or both ports active.

Fig. 14 shows the power for each module across its voltage range when active 100% of the time utilizing the same weightings and conditions as were used for energy measurements.

VI. Application Performance and Comparisons

Programming is accomplished by a multistep process. Individual programs are written, replicated, or generated in the C,

TABLE II
KILOCORE APPLICATION METRICS FOR OPERATION AT 1.1 V. *DOES NOT INCLUDE TIME SPENT WAITING FOR NETWORKS

Application	Cores	Branch Correct Predic.	Active Instr Per Cycle Per Core*	Instr Per Second (Billion instrs/s)	Throughput	Power (W)	Thruput/Watt
AES	974	100%	.93	802	21.4 Gbps	15.4	1.39 Gbps/W
LDPC 4095-bit 5-iterations	944	89%	.95	320	145.4 Mbps	8.21	17.7 Mbps/W
FFT 4096-point complex	980	94%	.98	407	823.9 MSamp/s	9.75	84.5 MSamp/s/W
Sort 100-Byte records	1000	96%	.96	131	2.12 GB/s	3.12	0.678 GB/s/W

C++, or Assembly language. An automatic mapping tool maps tasks to cores with considerations, such as avoiding faulty or partially functional processors; optimizations to take advantage of process, voltage, and temperature variations; self-healing for failures due to wear-out effects; and simultaneous execution of unrelated workloads.

Several applications have been mapped to KiloCore and their performance estimated using simulations, which assume custom chip packaging. Simulations are cycle accurate within a core, use subcycle precision for core interactions, fully model varied per-core frequencies, and utilize subinstruction energy measurements. Application code has been lightly to moderately optimized and additional effort would yield significant improvements. All four applications store instructions inside local processor memories, and so usage of independent memories, run-time instruction swapping, or run-time off-chip instruction streaming is not required. Performance metrics at a supply voltage of 1.1 V for applications are found in Table II.

An advanced encryption engine (AES) application is implemented with 974 processors. It uses 128-b keys and is organized into seven parallel lanes. At a reduced 0.9 V, it supports a throughput of 14.5 Gb/s while using 6.5 W. It is operable down to 560 mV, where a throughput of 1.23 Gb/s is achieved using 158 mW.

An LDPC decoder is implemented with 944 processors and 12 independent memories. It supports a (4095,3717) code with row and column weights of 64 and 6, and utilizes 12 parallel decoding lanes. At a reduced 0.9 V, with four decoding iterations, it has a throughput of 111 Mb/s while using 3.4 W. It is operable down to 760 mV, where it decodes 62 Mb/s using 1.1 W.

A 4096-point complex FFT application is implemented with 980 processors and 12 independent memories. It processes 16-b complex data and calculates 12 transforms in parallel. At 0.9 V, 567 MSamples/s are processed using 4.1 W. It is operable down to 760 mV, with 313 MSamples/s using 1.4 W. A second 4096-point complex FFT application was developed, which processes a single FFT transform at a time and uses 619 processors and 12 memories to transform 295 MSamples/s using 2.6 W at 0.9 V.

The first phase of an “external” record sort is implemented with 1000 processors; 100-B records contain a 10-B sorting key and are processed into sorted blocks of 185 KB in support of the second merging phase of the external sort. At 0.9 V, this application sorts 1.47 GB/s using 1.2 W. It is operable down to 560 mV, where it can sort 137 MB/s using 61 mW.

KiloCore’s applications are compared against a selection of Intel i7 and Nvidia GPU processors due to their wide acceptance and deployment, highly optimized hardware, and mature programming tools. In addition, Intel Core (and related Xeon) and Nvidia GPUs are frequently deployed in computing domains ranging from mobile, desktop, server, datacenter, to scientific supercomputer. Comparison data are given in Table III and include KiloCore data both unscaled and scaled to the same technology using data from Holt [2]. The AES comparisons on an Intel i7 [20] and Nvidia GPU [21] are taken from the literature, and do not use the specialized AES hardware present in many Intel processors. The LDPC comparisons on an i7 [22] and GPU [23] implement (9216,4608) and (2304,1152) codes with row and column weights of 6,3 and 24,12, respectively, and perform five decoding iterations. The i7 FFT uses the FFTW library with eight independent threads iterating on cached data. The GPU FFT is implemented on an Nvidia GTX 960 using the cufftExecC2C function from the Nvidia Cuda cuFFT library. Both implementations utilize single-precision floating-point operations. Interestingly, a hypothetical floating-point KiloCore would actually experience a speedup compared with this fixed-point version, which must explicitly handle data alignment and overflow functions. Sorting is implemented on an i7-3770k using std::sort in C++ with eight independent threads operating on separate record groups in cache, and is implemented on a GTX 960 using the sort function from the Nvidia Cuda Thrust library. Power is measured using on-die energy counters when available, or by the measured power delta with a correction for power supply efficiency. For cited designs without reported power, we use half of the thermal design power (TDP) [24]; i7 power numbers do not include uncore power. Area comparisons are made using die area, subtracting the estimated area for the graphics, memory controller, and unused cores in the comparison CPUs.

Across these applications and when scaled to the same fabrication technology, KiloCore at 1.1 V has geometric mean improvements of 4.3× higher throughput per area and 9.4× higher energy efficiency compared with the other processors.

TABLE III

APPLICATION METRICS AND COMPARISONS OF KiloCore WITH CPU AND GPU IMPLEMENTATIONS. KiloCore METRICS ARE NORMALIZED AGAINST THE COMPARISON DEVICE, ARE (COLUMNS 7–8) UNSCALED AND OPERATING AT 1.1 V, AND ARE †(COLUMNS 9–10) SCALED TO THE SAME TECHNOLOGY USING DATA FROM HOLT [2]. ‡ASSUMES DEVICE POWER IS HALF OF TDP [24]

Application	Device	Tech (nm)	Thruput	Thruput/ Watt	Thruput/ Area	KiloCore Relative Thruput/ Area	KiloCore Relative Thruput/ Watt	KiloCore† Relative Thruput/ Area	KiloCore† Relative Thruput/ Watt
AES	i7 920 [20] ‡	45	1.2 Gbps	0.018 Gbps/W	0.11 Gbps/mm^2	55.7	75.3	24.1	52.8
AES	Tesla C2050 [21] ‡	40	60 Gbps	0.50 Gbps/W	4.6 Mbps/mm^2	2.95	2.76	1.68	2.07
LDPC	i7 3960X [22] ‡	32	180 Mbps	2.77 Mbps/W	1.11 Mbps/mm^2	4.03	6.40	4.03	6.40
LDPC	GTX Titan [23] ‡	28	621.4 Mbps	4.97 Mbps/W	0.41 Mbps/mm^2	2.05	3.56	2.73	3.97
FFT	i7 3770k	22	1048 MSamp/s	18.4 MSamp/s/W	22.5 MSamp/s/ mm^2	1.17	4.58	2.71	6.65
FFT	GTX 960	28	5120 MSamp/s	76.4 MSamp/s/W	6.55 MSamp/s/ mm^2	0.57	1.11	0.76	1.23
Sort	i7 3770k	22	3.91 GB/s	0.074 GB/s/W	0.59 MB/s/mm^2	0.80	9.11	1.87	13.2
Sort	GTX 960	28	0.135 GB/s	0.004 GB/s/W	0.024 GB/s/mm^2	55.7	186	74.1	207

Significantly higher efficiencies are possible at lower supply voltages.

Analyzing programming effort, in general, is difficult; however, these benchmarks provided an opportunity to gain experience in scaling smaller application kernels into order-1000-core applications. Despite using an immature tool set, porting and upscaling all four applications to their final sizes of 944–1000 cores required in total one to two days of programming effort each.

VII. CONCLUSION

Processor maximum operating frequencies average 1.78 GHz at 1.10 V, which results in a maximum execution rate of 1.78 trillion MIMD instructions/s per chip. At a supply voltage of 0.84 V, 1000 cores process a maximum of 1.0 trillion instructions/s while dissipating 13.1 W. At a supply voltage of 0.56 V, processors dissipate 5.3 pJ per instruction at 115 MHz, which enables a chip to process 115 billion instructions/s while dissipating only 0.61 W; or multiple chips could execute 1.0 trillion instructions/s while dissipating only 5.3 W.

Throughout the history of computing, adding functionality onto a single chip has virtually always brought increased performance at a reduced cost [25]. While it is a near certainty that more chips containing 1000 processors will be built in the future [1], [26], there are challenging open questions regarding how those processors will communicate with each other and external system components, and how applications will be written for them. The KiloCore chip demonstrates the feasibility and some advantages of this promising new era.

REFERENCES

[1] K. Kim, "Silicon technologies and solutions for the data-driven world," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2015, pp. 1–7.

[2] W. M. Holt, "Moore's law: A path going forward," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Jan./Feb. 2016, pp. 8–13.

[3] L. T. Su, "Architecting the future through heterogeneous computing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2013, pp. 8–11.

[4] J. L. Gustafson, "Reevaluating Amdahl's law," *Commun. ACM*, vol. 31, no. 5, pp. 532–533, May 1988.

[5] M. J. Flynn, "Very high-speed computing systems," *Proc. IEEE*, vol. 54, no. 12, pp. 1901–1909, Dec. 1966.

[6] S. R. Vangal *et al.*, "An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, Jan. 2008.

[7] D. N. Truong *et al.*, "A 167-processor computational platform in 65 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 44, no. 4, pp. 1130–1144, Apr. 2009.

[8] S. Bell *et al.*, "TILE64 Processor: A 64-core SoC with mesh interconnect," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2008, pp. 88–89.

[9] M. Butts and A. M. Jones, "TeraOPS hardware: A new massively-parallel MIMD computing fabric IC," in *Proc. HotChips Symp. High-Perform. Chips*, Stanford, CA, USA, Aug. 2006, pp. 1–15.

[10] B. Bohnenstiehl *et al.*, "A 5.8 pJ/Op 115 billion ops/sec, to 1.78 trillion ops/sec 32nm 1000-processor array," in *Proc. Symp. VLSI Circuits*, Honolulu, HI, USA, Jun. 2016, pp. 1–2.

[11] B. Bohnenstiehl *et al.*, "KiloCore: A 32 nm 1000-processor array," in *Proc. HotChips Symp. High-Perform. Chips*, session 7, Aug. 2016.

[12] J. J. Pimentel and B. M. Baas, "Hybrid floating-point modules with low area overhead on a fine-grained processing core," in *Proc. IEEE Asilomar Conf. Signals, Syst. Comput. (ACSSC)*, Pacific Grove, CA, USA, Nov. 2014, pp. 1–2.

[13] A. Stillmaker, L. Stillmaker, and B. Baas, "Fine-grained energy-efficient sorting on a many-core processor array," in *Proc. IEEE 18th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Singapore, Dec. 2012, pp. 652–659.

[14] Z. Yu and B. M. Baas, "A low-area multi-link interconnect architecture for GALS chip multiprocessors," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 5, pp. 750–762, May 2010.

[15] A. T. Tran and B. M. Baas, "Achieving high-performance on-chip networks with shared-buffer routers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 6, pp. 1391–1403, Jun. 2014.

[16] R. W. Apperson, Z. Yu, M. J. Meeuwsen, T. Mohsenin, and B. M. Baas, "A scalable dual-clock FIFO for data transfers between arbitrary and haltable clock domains," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 10, pp. 1125–1134, Oct. 2007.

[17] Z. Yu *et al.*, "AsAP: An asynchronous array of simple processors," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 695–705, Mar. 2008.

[18] Z. Yu and B. M. Baas, "High performance, energy efficiency, and scalability with GALS chip multiprocessors," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 1, pp. 66–79, Jan. 2009.

[19] D. M. Chapiro, "Globally-asynchronous locally-synchronous systems," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 1984.

[20] K. Iwai, T. Kurokawa, and N. Nisikawa, "AES encryption implementation on CUDA GPU and its analysis," in *Proc. 1st Int. Conf. Netw. Comput. (ICNC)*, Nov. 2010, pp. 209–214.

[21] Q. Li, C. Zhong, K. Zhao, X. Mei, and X. Chu, "Implementation and analysis of AES encryption on GPU," in *Proc. IEEE 14th Int. Conf. High Perform. Comput. Commun., IEEE 9th Int. Conf. Embedded Softw. Syst. (HPCC-ICESS)*, Jun. 2012, pp. 843–848.

[22] X. Pan, X.-F. Lu, M.-Q. Li, and R.-F. Song, "A high throughput LDPC decoder in CMMB based on virtual radio," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Apr. 2013, pp. 95–99.

[23] G. Wang, M. Wu, B. Yin, and J. R. Cavallaro, "High throughput low latency LDPC decoding on GPU for SDR systems," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Dec. 2013, pp. 1258–1261.

[24] M. Butler, "'AMD Bulldozer Core—A new approach to multithreaded compute performance for maximum efficiency and throughput," in *Proc. IEEE HotChips Symp. High-Perform. Chips (HotChips)*, session 7, Aug. 2010.

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

[26] S. Borkar, "Thousand core chips: A technology perspective," in *Proc. 44th Annu. Design Autom. Conf. (DAC)*, New York, NY, USA, 2007, pp. 746–749.

Brent Bohnenstiehl (S'15) received the B.S. degree (Hons.) in electrical engineering from the University of California, Davis, CA, USA, in 2006, where he is currently pursuing the Ph.D. degree in electrical and computer engineering.

He has been a Graduate Student Researcher with the VLSI Computation Laboratory, Davis, since 2011. His current research interests include processor architecture, VLSI design, hardware-software codesign, dynamic voltage and frequency scaling algorithms and circuits, and many-core compilers and other programming and simulation tools.

Aaron Stillmaker (S'09–M'17) received the B.S. (*magna cum laude*) degree in computer engineering from the California State University, Fresno, CA, USA, in 2008, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of California (UC Davis), Davis, CA, USA, in 2013 and 2015, respectively.

In 2013, he was an Intern with the Circuit Research Lab, Intel Labs, Hillsboro, OR, USA. From 2008 to 2015, he was a Graduate Student Researcher with the VLSI Computation Laboratory, UC Davis. Since 2017, he has been an Assistant Professor with the Electrical and Computer Engineering Department, California State University at Fresno, Fresno, CA, USA. His current research interests include manycore processor architecture, many-core applications, and VLSI design.

Dr. Stillmaker was a GAANN Fellow from 2008 to 2015. He is a member of Eta Kappa Nu and Tau Beta Pi. He received the ECEGP Fellowship in 2014 and 2015 and the Dissertation Quarter Fellowship in 2015. He was a President's Scholar from 2004 to 2008.

Jon J. Pimentel (S'07) received both the B.S. degree (Hons.) in electrical engineering in 2009 and the M.S. degree in electrical and computer engineering from the University of California, Davis, CA, USA, in 2015, where he is currently pursuing the Ph.D. degree in electrical and computer engineering.

He has been a Graduate Student Researcher with the VLSI Computation Laboratory, University of California at Davis, since 2009. In 2013, he was a Graduate Technical Intern with the Many Integrated Core Group, Intel, Hillsboro, OR, USA. His current research interests include floating-point architectures, VLSI design, synthetic aperture radar imaging, scientific applications, and many-core processor architecture.

Mr. Pimentel has been a GAANN Fellow since 2009. He received the Best Student Paper Award Third Place at Asilomar 2014. He received the Graduate Research Mentorship Fellowship in 2011, the UCD and Humanities Graduate Research Award in 2012 and 2014, the Frank and Carolan Walker Fellowship in 2012 and 2014, the George S. and Marjorie Butler Fellowship in 2014, the ECEGP Fellowship in 2014 and 2015, the ECE TA Program Support Fellowship in 2015, and the Herbert Tryon Fellowship, the Laura Perrot Mahan Fellowship, and the Dissertation Writing Fellowship in 2016.

Timothy Andreas (S'13) received the B.S. degree (Hons.) in electrical engineering from the University of California, Davis, CA, USA, in 2013, where he is currently pursuing the Ph.D. degree in electrical and computer engineering.

His research interests include energy-efficient and high-performance machine learning algorithms and processor architectures.

Bin Liu (S'08) received the B.S. degree in information engineering from Shanghai Jiao Tong University, Shanghai, China, in 2007, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of California (UC Davis), Davis, CA, USA, in 2010 and 2016, respectively.

From 2008 to 2014, he was a Graduate Student Researcher with the VLSI Computation Laboratory, UC Davis. From 2014 to 2016, he was a Senior Software Engineer with the GPU Performance Team, AMD, Sunnyvale, CA, USA. Since 2016, he has been a Software Engineer with the Machine Learning and Risk Team, Uber, San Francisco, CA, USA. His current research interests include high-performance many-core processor architecture, dynamic supply voltage and frequency scaling algorithms and circuits, and parallel encryption engine implementations.

Dr. Liu received the Best Student Paper Award third place at the IEEE MWSCAS 2015 and the Best Student Paper nomination at the IEEE Asilomar 2011.

Anh T. Tran (S'07–M'12) received the B.S. degree (Hons.) in electronics engineering from the Posts and Telecommunications Institute of Technology, HCMC, Vietnam, in 2003, and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Davis, CA, USA, in 2009 and 2012, respectively.

He is currently a Lead Hardware Research Engineer with Cavium Inc, San Jose, CA, USA, where he is participating in the development of software defined network ASICs for enterprise and data center network switches. From 2012 to 2014, he was with Xpliant Inc., an ASIC startup in San Jose, which was later acquired by Cavium Inc. in 2014. His current research interests include VLSI designs, multicore architectures, onchip interconnects, and reconfigurable systems on chip.

Dr. Tran was a VEF Fellow from 2006 to 2012. He was a recipient of the Best Paper Award at the IEEE ICCD conference for his work on high performance on-chip router designs in 2011. He received the UC Davis College of Engineering Zuhair A. Munir Award for Best Doctoral Dissertation Honorable Mention in 2013.

Emmanuel Adeagbo (S'07) received the B.S. degree in electrical engineering and computer science from the University of California at Berkeley, Berkeley, CA, USA, in 2009, and the M.S. degree in electrical engineering from the University of California, Davis, CA, USA, in 2016. His thesis was on energy-efficient pattern matching methods on a fine-grained many-core platform.

He is currently a Physical Design Engineer with the Intel Platform Engineering Group's Big Core division. His research interests include energy-efficient regular expression applications, VLSI, and digital architecture design.

Bevan M. Baas (M'95–SM'11) received the B.S. degree in electronic engineering from California Polytechnic State University, San Luis Obispo, CA, USA, in 1987, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 1990 and 1999, respectively.

From 1987 to 1989, he was with Hewlett-Packard, Cupertino, CA, USA. In 1999, he joined Atheros Communications, Santa Clara, CA, USA, as an early employee and a core member of the team which developed the first commercial IEEE 802.11a Wi-Fi wireless LAN solution. In 2003, he joined the Department of Electrical and Computer Engineering, University of California, Davis, CA, USA, where he is currently a Professor. He leads projects in architectures, hardware, applications, and software tools for VLSI computation.

Dr. Baas was an NSF Fellow from 1990 to 1993 and a NASA GSR Fellow from 1993 to 1996. He received the National Science Foundation CAREER Award in 2006, the Best Paper Award at ICCD 2011, the Best Student Paper Award third place at the IEEE Asilomar 2014, the Best Student Paper Award third place at the IEEE MWSCAS 2015, WACIest Best-In-Session Paper at DAC 2010, and the Best Paper nominations at the IEEE Asilomar 2011 and the IEEE BioCAS 2010. He supervised the research that earned the College of Engineering Award for Best Doctoral Dissertation Honorable Mention in 2013, and received the Most Promising Engineer/Scientist Award by AISES in 2006. He is currently an Associate Editor of the IEEE TRANSACTIONS ON VLSI SYSTEMS and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II. From 2007 to 2012, he was an Associate Editor of the IEEE JOURNAL OF SOLID-STATE CIRCUITS. He has served as a Guest Editor of Special Issues of IEEE Micro and IEEE Design & Test of Computers. He served as the Co-Chair of HotChips 2011, the Track Co-Chair of the IEEE DSC 2017, the Co-Chair of the DAC PAPA Workshop 2011, and a Program Committee Member of numerous conferences.