The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology

Florian Zaruba, *Student Member, IEEE*, and Luca Benini, *Fellow, IEEE*

***Abstract*— The open-source RISC-V instruction set architecture (ISA) is gaining traction, both in industry and academia. The ISA is designed to scale from microcontrollers to server-class processors. Furthermore, openness promotes the availability of various open-source and commercial implementations. Our main contribution in this paper is a thorough power, performance, and efficiency analysis of the RISC-V ISA targeting baseline "application class" functionality, i.e., supporting the Linux OS and its application environment based on our open-source single-issue in-order implementation of the 64-bit ISA variant (RV64GC) called Ariane. Our analysis is based on a detailed power and efficiency analysis of the RISC-V ISA extracted from silicon measurements and calibrated simulation of an Ariane instance (RV64IMC) taped-out in GlobalFoundries 22 FDX technology. Ariane runs at up to 1.7-GHz, achieves up to 40-Gop/sW energy efficiency, which is superior to similar cores presented in the literature. We provide insight into the interplay between functionality required for the application-class execution (e.g., virtual memory, caches, and multiple modes of privileged operation) and energy cost. We also compare Ariane with RISCY, a simpler and a slower microcontroller-class core. Our analysis confirms that supporting application-class execution implies a nonnegligible energy-efficiency loss and that compute performance is more cost-effectively boosted by instruction extensions (e.g., packed SIMD) rather than the high-frequency operation.**

***Index Terms*— Architectural analysis, cost application-class, RISCV.**

I. Introduction

THE relatively new open instruction set architecture (ISA) RISC-V [1] has already seen a widespread adoption in industry and academia [2]–[5]. It is based on reduced instruction set computer (RISC) principles and heavily relies on standard and nonstandard extensions to tailor the ISA without cluttering the instruction set base. Furthermore, the base ISA is split into privileged and nonprivileged instructions. While the nonprivileged ISA governs the base instruction set, the privileged ISA includes different levels of hardware support needed to run an operating system (OS). The common and standardized RISC-V extensions are integer (I), multiply/divide support (M), atomic memory operations (A), and single (F) and double (D) precision IEEE floating point support. Together they form the general-purpose processing extension (G). Furthermore, the RISC-V ISA supports variable length instructions. Currently, it only defines 32- and 16-bit compressed instructions (C).

The instruction set was designed to scale from microcontrollers to server-class platforms. While there already exist a plethora of open microcontroller cores [6], [7], there are fewer cores available in the higher, Linux-capable, performance range, mostly due to the increased design and verification costs. CPUs, which offer support for UNIX-like OSs, are usually called application-class processors. The hardware overhead to efficiently support OSs like Linux is significant: To enable fast address translation, a translation lookaside buffer (TLB) and a page table walker (PTW) are needed. A Linux-like OS needs at least a few dozen megabytes (MB) of main memory. In most of the cases, this memory will be off-chip making it inefficient to be accessed constantly and requiring some sort of caching mechanism. Cache lookup and address translation are often on the critical path in modern CPU designs as the accessing memory is slow. Still, keeping the operating frequency high is of importance since OSs contain large portions of highly serial code. This requires further pipelining and more sophisticated hardware techniques to hide the increased pipeline depth such as scoreboarding, branch prediction, and more elaborate out-of-order techniques [8], [9]. This increases both static power and dynamic power. Furthermore, the advent of symmetric multiprocessing (SMP) support in OSs made it necessary to provide efficient and fast atomic shared memory primitives in the ISA. RISC-V provides this as a part of the A-extension in the form of load reserve and store conditional as well as atomic fetch-and-op instructions that can perform operations such as integer arithmetic, swapping, and bitwise operations close to memory.

Nevertheless, there are significant gains in having support for a full-blown OS: the OS eases programmability by providing a standardized interface to user programs, memory management, isolation between user programs, and a vast

Manuscript received January 25, 2019; revised May 23, 2019; accepted June 26, 2019. Date of publication July 26, 2019; date of current version October 23, 2019. This work was supported by the European Union's Horizon 2020 Research and Innovation Program through the Project "OPRECOMP" under Grant 732631. *(Corresponding author: Florian Zaruba.)*

F. Zaruba is with the Integrated Systems Laboratory, ETH Zürich, 8092 Zürich, Switzerland (e-mail: zarubaf@iis.ee.ethz.ch).

L. Benini is with the Integrated Systems Laboratory, ETH Zürich, 8092 Zürich, Switzerland, and also with the Department of Electrical, Electronic and Information Engineering (DEI), University of Bologna, 40126 Bologna, Italy (e-mail: lbenini@iis.ee.ethz.ch).

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

Digital Object Identifier 10.1109/TVLSI.2019.2926114

amount of libraries, drivers, and user programs. Furthermore, full-featured OSs (e.g., Sel4 [10]) provide an additional guaranteed layer of security.

Energy efficiency is becoming the paramount design goal for the next generation architectures [11], [12]. Due to its modularity, the RISC-V instruction set offers a wide variety of microarchitectural design choices, ranging from low-cost microcontroller to high performance, out-of-order, server-class CPUs supporting Linux [13]. This modularity makes the ISA suitable for a more thorough analysis of different architectural features and their impact on energy efficiency.

Our work aims at providing insight on the energy cost and design tradeoffs involved in designing a RISC-V core with the support for a full-featured OS. For a thorough analysis, we designed a competitive 64-bit, six-stage, in-order application-class core that has been manufactured in GLOBALFOUNDRIES 22 FDX technology. We performed extensive silicon characterization and a detailed, per-unit efficiency analysis based on silicon-calibrated postlayout simulations. The core runs at 1.7 GHz and achieves an efficiency of up to 40 Gop/sW.

In particular, our main contributions in this paper are the following.

1) Implementation of an in-order (out-of-order execute, in-order commit), single-issue, 64-bit application-class processor called Ariane. Ariane has been open-sourced on GitHub.[1]
2) Silicon integration of Ariane in a state-of-the-art 22-nm SOI process.
3) Exploration of tradeoffs in performance and efficiency based on silicon measurements.
4) Thorough analysis of the RISC-V ISA and Ariane's microarchitecture based on measurements and silicon-calibrated postlayout simulations.

We explore Ariane's microarchitecture in detail in Section II. In Section III, we touch upon physical design of a particular Ariane instance. Finally, Section IV contains a detailed power, performance, and efficiency analysis.

A. Related Work

Although RISC-V is a relatively young ISA, there already exists a plethora of different commercial and open-source microarchitectural implementations. Currently, most of them focus on the base integer subset and do not feature more sophisticated application-class features, such as virtual memory (VM). To the best of our knowledge, this is the first study, backed by silicon measurements, focusing on the energy breakdown across the various functional units and on the design rationale for microarchitectural features, with links to ISA requirements on a competitive 64-bit application-class core.

A well-known, mature, and competitive application-class implementation of the RISC-V ISA is the one obtained through the Rocketchip generator. Rocketchip a parametric system on chip (SoC) generator, capable of producing instances ranging in complexity between single application-class cores and chache-coherent multicore systems [14]. Rocketchip is written in Chisel—a hardware domain-specific language embedded in Scala. The generator allows for instantiating either "Rocket," a single-issue, in-order core or an out-of-order, superscalar core called BOOM. Another major effort to provide a variety of different open-source RISC-V cores is the SHAKTI project [15] led by IIT Madras. One particular implementation that is similar to Ariane is the 64-bit, five-stage, in-order C-class core. We provide a detailed comparison between Ariane, Rocket, BOOM, and SHAKTI C-class in Table I and related discussion in Section IV-D.

[1] https://github.com/pulp-platform/ariane

TABLE I
RISC-V CORE COMPARISONS

	Ariane	**Rocket [14]**	**Boom 2-w [27]**	**Shakti [28]**
Bits	64	32/64	64	64
User Spec	IMC	IMAFDC	IMAFD	IMAFD
Priv. Spec	1.11	1.11	1.11	1.10
Tech	GF 22 nm	TSMC 45 nm	TSMC 45 nm	22 nm
Speed	1.7 GHz	1.6 GHz [29]	1.5 GHz [27]	800 MHz
Area	$0.3\,\mathrm{mm}^2$	$0.5\,\mathrm{mm}^2$	$1.7\,\mathrm{mm}^2$ [27]	$0.29\,\mathrm{mm}^{2*}$
Power	52 mW	125 mW [30]	300 mW [30]	90 mW
IPC	0.87[a]	0.95‡	1.45‡	0.9[1]
Energy/Op	52 pJ	100 pJ [30]	133 pJ [30]	122 pJ

* Assumption: 1 GE = $0.199\,\mu\mathrm{m}^2$ and the same cache configuration as our work.
[a] Measured on Dhrystone benchmark compiled with riscv64-unknown-elf-gcc 7.2.0 using recommended ARM compiler settings [31].
‡ Measured on Coremark [17] benchmark with unknown compiler settings.
[1] Measured on Dhrystone benchmark with unknown compiler settings.

II. ARCHITECTURE

Ariane is a 64-bit, single, in-order issue (out-of-order execute) RISC-V core. It has support for hardware multiply/divide, atomic memory operations, as well as an IEEE compliant floating point unit (FPU). Furthermore, it supports the compressed instruction set extension as well as the full privileged instruction set extension. It implements a 39-bit, page-based VM scheme (SV39). The primary design goal of the microarchitecture was to reduce critical path length while keeping instructions per cycle (IPC) losses moderate. The target logic depth was chosen to be below 30 NAND gate equivalent (GE), which is just a factor of two higher than the state-of-the-art, highly tuned, server-class, out-of-order processors [16]. To achieve desired performance goals, a synthesis-driven design approach leads to a six-stage pipelined design. To reduce the penalty of branches, the microarchitecture features a branch predictor. A high-level block diagram is depicted in Fig. 1. In the following, we give a stage-by-stage description of the complete microarchitecture.

1) *PC Generation*: It is responsible for selecting the next program counter (PC). This can either come from control and status registers (CSRs) when returning from an exception, the debug interface, and a mispredicted branch, or a consecutive fetch.
2) *Instruction Fetch*: It contains the instruction cache, the fetch logic, and the predecode logic that guides the branch prediction of the PC stage.

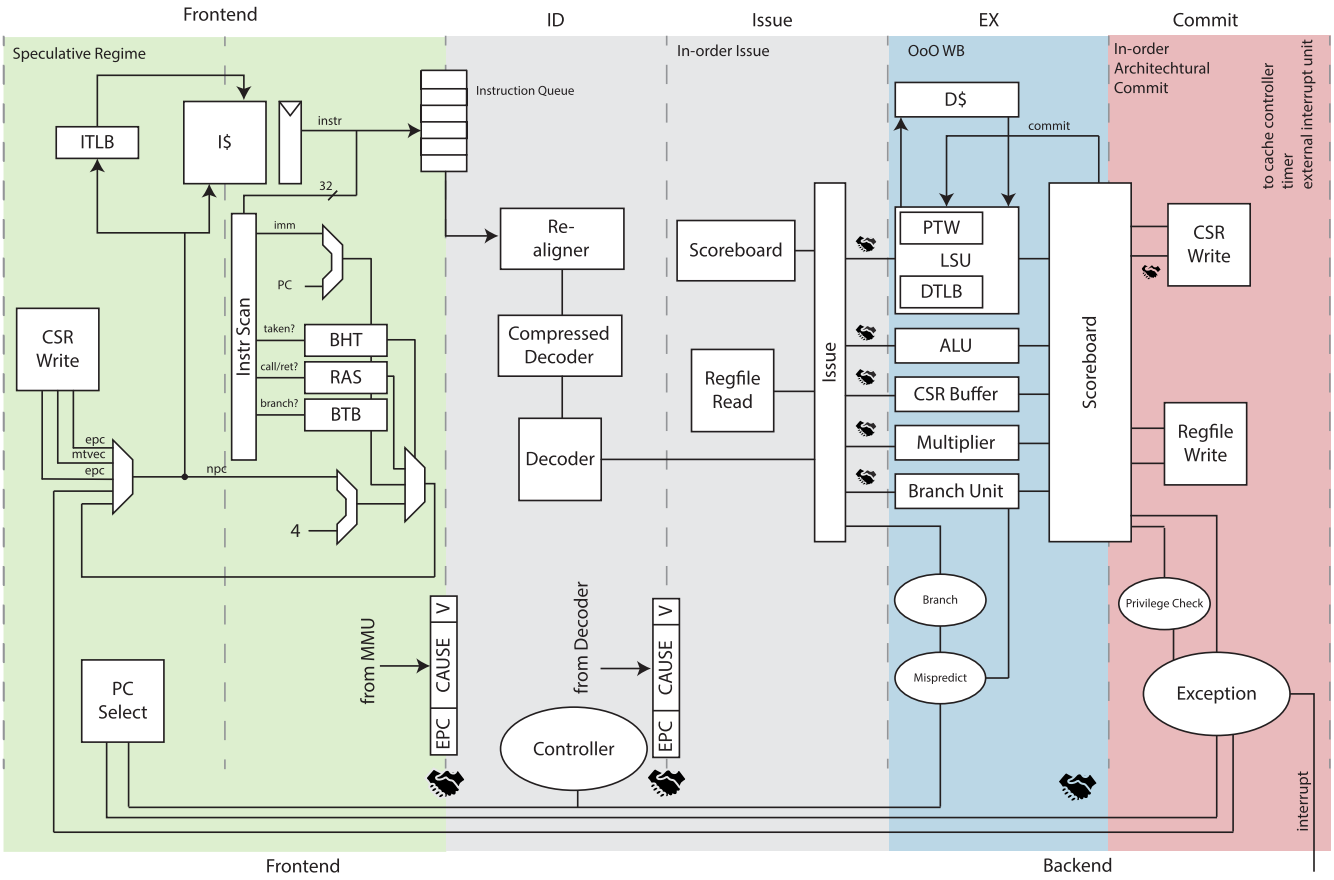

Fig. 1. Block diagram of Ariane; depicted are the six separate stages that exchange the data via handshaking. The scoreboard logically wraps the execute stage (see Section II-F) and provides an interface for the issue and commit stages. For a detailed description of each pipeline stage and functional unit, see Section II.

TABLE II
RISC-V STATIC BINARY ANALYSIS (RISCV64-UNKNOWN-LINUX-GNU-GCC 7.2.0)

Benchmark [17]	Compr. Ratio	Branches [%]	Calls [%]
cjpeg-rose7	0.71	34.78	6.71
dhrystone	0.72	35.36	6.78
linear-alg-mid-100x100-sp	0.72	33.65	6.17
loops-all-mid-10k-sp	0.72	33.78	6.22
nnet	0.72	33.75	6.17
parser-125k	0.72	33.73	6.43
radix2-big-64k	0.85	22.77	1.99
sha	0.72	33.25	6.01
zip	0.72	33.48	6.21

a) Address Translation: The instruction cache is virtually indexed and physically tagged and fully parametrizable. The PC calculated by the previous stage is split into page offset (lower 12 bit) and virtual page number (bit 12 up to 39). The page offset is used to index into the instruction cache in the first cycle, while the virtual page number is simultaneously used for address translation through the instruction TLB. In case of a TLB miss, the cache pipeline is stalled until the translation is valid.

b) Cache Pipelining: The data coming from the instruction cache's data arrays are registered before being predecoded to mitigate the effects of the long propagation delay of the memory macros in the cache. This has the side effect that even on a correct control flow prediction, we will lose a cycle as we cannot calculate the next PC in the same cycle we receive the data from the instruction cache. With the additional compressed instruction set extension, this is usually not a problem as (with a fetch width of 32 bit) we are fetching 1.5 instructions on average, and approximately 70% of all instructions are compressed. Furthermore, approximately 35% of the instructions are branches (see Table II). This means we can easily tolerate a single cycle delay on branch prediction (caused by the additional register stage after the instruction cache) and still generate a consecutive instruction stream for the processor's back-end. This is possible as, due to compressed instructions, we are fetching up to two instructions and can use the stall cycle to execute the other instruction.

c) Front End: Together with the PC stage, the instruction fetch stage forms the processor's front end. The front end is fully decoupled from the back-end, which is in charge of executing the instruction stream. The decoupling is implemented as a FIFO of configurable depth. Instructions are stored in the compressed form in the queue minimizing the number of flip-flops necessary for the instruction FIFO. Mis-predicted control flow instructions are resolved during the execute stage in a specialized branch unit [18].

3) *Instruction Decode*: It realigns potentially unaligned instructions, decompresses them and decodes them. Decoded instructions are then put into an issue queue in the issue stage.
4) *Issue Stage*: It contains the issue queue, a scoreboard, and a small reorder buffer (ROB). Once all operands are ready, the instruction is issued to the execute stage. Dependencies are tracked in the scoreboard and operands are forwarded from the ROB if necessary.
5) *Execute Stage*: It houses all functional units. Every functional unit is handshaked and readiness is taken into account during the instruction issue. Furthermore, we distinguish between fixed and variable latency units. Fixed latency is the integer arithmetic logic unit (ALU), multiplier/divider, and CSR handling. The only variable latency units are currently the FPU and the load/store unit (LSU). Instructions can retire out-of-order from the functional units. Write-back conflicts are resolved through the ROB.
6) *Commit Stage*: It reads from the ROB and commits all instructions. Instructions ready to commit are brought into issue order by a small ROB to enable precise exceptions. Stores and atomic memory operations are held in a store buffer until the commit stage confirms their architectural commit. Finally, the register file is updated by the retiring instruction. To avoid artificial starvation because of a full ROB, the commit stage can commit two IPC.

Next, we describe the main units of the microarchitecture, summarizing key features.

A. Branch Prediction

As the pipeline depth of processors increases, the cost for mis-predicted branches rises significantly. Mis-prediction can occur on the jump target address (the jump address is determined by a register value) as well as on a mis-predicted branch outcome. On misprediction, the front end as well as the decode and issue stages need to be flushed, which introduces at least a five-cycle latency in the pipeline, and even more on TLB or instruction cache misses.

To mitigate the negative impact of control flow stalls on IPC, Ariane contains three different means of predicting the next PC, namely, a branch history table (BHT), branch target buffer (BTB), and return address stack (RAS). To facilitate branch-prediction, Ariane does a light predecoding in its fetch interface to detect branches and jumps. It furthermore needs to realign instruction fetches as interleaved compressed instructions (16-bit instructions) can offset regular (32-bit) instructions effectively making it possible for an instruction to wrap the 32-bit fetch boundary.

A classic two-bit saturation counter BHT is used for predicting on the branch outcome. Branches in RISC-V can only jump relative to the PC, which makes it possible to redirect control flow immediately. If there is no valid prediction available, the static prediction is being used as a fall-back strategy. Static prediction in RISC-V is defined as backward jumps (negative immediate) always taken and forward jumps (positive immediate) never taken, hence they can be decided very efficiently by looking at a single bit in the immediate field. Furthermore, the ISA provides PC-relative and absolutely addressed control flow changes. PC-relative unconditional jumps can be resolved as soon as the instruction is being fetched. Register-absolute jumps can either be predicted using the BTB or the RAS depending on whether the jump is used as a function call or not.

B. Virtual Memory

To support an OS, Ariane features full hardware support for address translation via a memory management unit (MMU). It has separate, configurable data, and instruction TLBs. The TLBs are fully set-associative, flip-flop-based, standard-cell memories. On each instruction and data access, they are checked for a valid address translation. If none exists, Ariane's hardware PTW queries the main memory for a valid address translation. The replacement strategy of TLB entries is pseudo least recently used (PLRU) [19].

C. Exception Handling

Exceptions can occur throughout the pipeline and are, hence, linked to a particular instruction. The first exception can occur during the instruction fetch when the PTW detects an illegal TLB entry. During decode, illegal instruction exceptions can occur while the LSU can also fault on address translation or trigger an illegal access exception. As soon as an exception has occurred, the corresponding instruction is marked and auxiliary information is saved. When the exception instruction finally retires, the commit stage redirects the instruction front end to the exception handler.

Interrupts are asynchronous exceptions, which are synchronized to a particular instruction. This results in the commit stage waiting for a valid instruction to retire, in order to take an external interrupt and associating an exception with it. Atomic memory operations must not be interrupted, which simply translates to not taking an interrupt when we retire an atomic instruction. The same holds true for atomic CSR instructions which can alter the hart's (HARdware Thread) architectural state.

D. Privileged Extensions

In addition to the VM, the privileged specification defines more CSRs that govern the execution mode of the hart. The base supervisor ISA defines an additional interrupt stack for supervisor mode interrupts as well as a restricted view of machine mode CSRs. Access to these registers is restricted to the same or a higher privilege level.

CSR accesses are executed in the commit stage and are never done speculatively. Furthermore, a CSR access can have side effects on subsequent instructions which are already in the pipeline and have been speculatively executed, e.g., altering the address translation infrastructure. This makes it necessary to completely flush the pipeline on such accesses.

In addition to the base ISA, the privileged ISA defines a handful more instructions ranging from power hints (sleep and wait for interrupt) to changing privilege levels (call to the supervising environment as well as returning). As those

instructions alter the CSR state as well as the privilege level, they are only executed nonspeculatively in the commit stage.

The RISC-V ISA defines separate memory streams for instruction, data, and address translation, all of which need to be separately synchronized with special memory ordering instructions (fences). For caches, this means that they are either coherent or need to be entirely flushed. As the TLBs in Ariane are designed with flip-flops, they can be efficiently flushed in a single cycle.

E. Register Files

The core provides two physically different register files for floating-point and integer registers. We provide the choice of either a latch-based or flip-flop-based implementation. The advantage of the latch-based approach is that it is approximately half the area of the flip-flop version. A known duty cycle is of importance when using a latch-based register file as capturing is happening on the falling edge of the clock. Therefore, (high-speed) clock generators need to take care of a balanced and low jitter duty cycle.

F. Scoreboard/Reorder Buffer

The scoreboard, including the ROB, is implemented as a circular buffer that logically sits between the issue and execute stage and contains as the following.

1) Issued, decoded, in-flight instructions that are currently being executed in the execute stage. Source and destination registers are tracked and checked by the issue stage to track data hazards. As soon as a new instruction is issued, it is registered within the scoreboard.
2) Speculative results written back by the various functional units. As the destination register of each instruction is known, the results are forwarded to the issue stage when necessary. The commit stage reads finished instructions and retires them, therefore making room for new instructions to enter the scoreboard.

Write after write (WAW) hazards in the scoreboard are resolved through a lightweight renaming scheme, which increases the logical register address space by one bit. Each issued instruction toggles the MSB of its destination register address and subsequent read addresses are renamed to read from the latest register address.

G. Functional Units

Ariane contains six functional units as follows.

1) *ALU:* Covers most of the RISC-V base ISA, including branch target calculation.
2) *LSU:* Manages integer and floating-point load/stores as well as atomic memory operations. The LSU interfaces to the data cache using three master interfaces. One dedicated for the PTW, one for the load unit, while the last one is allocated to the store unit. The data cache is a parameterizable write-back cache that supports hit-undermiss functionality on different master ports. In addition, the store unit contains a variable-size store buffer to hide the store latency of the data cache. Ideally, the store-buffer is sized so that context store routines commonly found in OS code can be retired with an IPC of 1.
3) *FPU:* Ariane contains an IEEE compliant floating-point unit with custom trans-precision extensions [20], [21]
4) The branch unit is an extension to the ALU which handles branch prediction and branch correction.
5) *CSR:* RISC-V mandates atomic operations on its CSR, as they need to operate on the most up-to-date value Ariane defers reading or writing until the instruction is committed in the commit stage. The corresponding write data are buffered in this functional unit and read again when the instruction is retiring.
6) *Multiplier/Divider:* This functional unit houses the necessary hardware support for the M-extension. The multiplier is a fully pipelined two-stage multiplier. We rely on retiming to move the pipeline register into the combinational logic during synthesis. The divider is a bit-serial divider with input preparation. Depending on the operand values, division can take from 2 to 64 cycles.

H. Caches

Both data and instruction caches are virtually indexed and physically tagged. Set-associativity and cache-line size of both caches can be adapted to meet core area constraints and timing. As even fast cache memories are relatively slow compared to logic, the instruction cache and data cache both have an additional pipeline stage on their outputs. This allows for relatively easy path balancing by means of deskewing (i.e., adjusting the memories clock to arrive earlier or later than the surrounding logic).

I. Memory and Control Interfaces

The core contains a single Advanced eXtensible Interface (AXI) five-master port as well as four interrupt sources as follows.

1) *Machine External Interrupts:* Machine-mode platform interrupts, e.g., Universal Asynchronous Receiver Transmitter (UART), Serial Peripheral Interface (SPI), and so on.
2) *Supervisor External Interrupts:* Supervisor-mode platform interrupts (i.e., the OS is in full control)
3) *Machine Timer Interrupt:* Platform timer (part of the RISC-V privileged specification). Used by the OS for time-keeping and scheduling.
4) *Machine Software Interrupt:* Interprocessor interrupts.

The master port is arbitrated between instruction fetch, data cache refill, and write-back as well as cache bypass (i.e., uncached) accesses.

J. Debug Interface

Ariane contains a RISC-V compliant debug interface [22]. For the implementation of the debug interface, an execution-based approach was chosen to keep the debug infrastructure minimally invasive on the microarchitecture and, therefore, improving on efficiency and critical path length (e.g., no multiplexers on CSR or general-purpose registers). The core uses its existing capability to execute instructions to facilitate

debugging by fetching instructions from a debug RAM. To put the core into debug mode, an interruptlike signal is asserted by the debug controller and the core will jump to the base address of the debug RAM. Only one additional instruction (dret) is required to return from debug mode and continue execution. Communication with the external debugger is done through a debug module (DM) peripheral situated in the peripheral clock and power domain.

K. Tracing and Performance Counters

We support extensive tracing in register transfer level (RTL) simulation. All register and memory accesses are traced together with their (physical and virtual) addresses and current values. Currently, we do not support PC tracing in hardware.

Performance counters are mapped into the CSR address space. In particular, we support counting the following events: cycle count, instructions-retired, L1 instruction/data cache miss, instruction/data TLB miss, load/store instruction counter, exception counter, and various branch-prediction metrics.

L. Application-Class Features

In comparison to nonapplication-class cores, there is a significant overhead attached to supporting a full-fledged OS (e.g., Linux). In particular, multiple layers of trusted and less trusted execution modes are needed to support the concept of privilege levels. This inflates the architectural state by requiring an interrupt stack and CSRs replication as well as managing access permissions to given architectural features. Furthermore, the essential requirement of supporting VM requires address translation on the critical paths to and from instruction and data memory which further increases the energy costs and, hence, impacts energy efficiency to access memory.

M. Implementation Remarks

The design of a fast processor with a reasonably high IPC poses some interesting challenges. Significant complexity revolves around the L1 memory interface which ideally should be large, low latency, and fast (running at the speed of the core's logic). The introduction of VM adds to the already existing complexity. Large portions of the design are built around the idea to hide (especially the load) memory latency by cleverly scheduling other instructions and trying to do as much useful work as possible in each clock cycle. Furthermore, the introduction of more privileged architectural states in the form of virtual to physical address translation as well as additional CSRs results in an increased area and (leakage) power. These cost factors are analyzed in detail in Section IV.

III. Implementation

Ariane has been taped-out in GLOBALFOUNDRIES 22 FDX. The coreplex has been hardened as a macro and uses a shared SoC infrastructure for off-chip communication [23]. The Ariane coreplex can communicate with the SoC via a full-duplex 64-bit data and address AXI interconnect. The SoC contains 520 kB of on-chip scratchpad memory and an extensive set of peripherals such as HyperRAM, SPI, UART, and I2C. The core is separately clocked by a dedicated frequency locked loop (FLL) in the SoC domain.

TABLE III
ARCHITECTURAL DESIGN CHOICES FOR SILICON IMPLEMENTATION

Parameter	Chosen
BHT	8
BTB	8
ROB Entries	8
RAS Entries	0
Fetch latency	1
L1 I-cache (4-way) size	16 kB
L1 D-cache (8-way) size	32 kB
L1 D-cache latency	3
Integer ALU latency	1
Register File	31x64 flip-flops

The Ariane coreplex can be separately supplied and powered down. Furthermore, the logic cells can be forward body biased (FBB) to increase the speed at the expense of leakage power. The taped-out instance of Ariane contains 16 kB of instruction cache with a set-associativity of four. The data cache is 32 kB in size with a set-associativity of eight. The instruction and data TLB are each 16 entries in size. The architectural parameter settings are listed in Table III. We present an analysis of the impact of architectural choices on IPC in Fig. 2. Note, however, that at the time of tape-out, the RAS was not present and that BTB and BHT have been a combined data structures which could either be used for predicting on the address or on a branch outcome. A single entry of the BTB comprises of a valid bit and a 64-bit jump target address while the BHT only contains a valid flag and a two-bit saturation counter. Hence, the size of this combined data structure was limited by the amount of BTB entries. This limitation was alleviated after the tape-out.

The architectural choices have further been influenced by physical design considerations. To keep IPC losses moderate, the ALU was architected to be single cycle. The ROB, which comprises of a content addressable memory (CAM), is placed and routed using an unguided, standard cell-based, and digital design flow. As a CAM requires comparators and multiplexers for each entry, there is a natural limit to what extend this CAM can be automatically placed and routed before having a severe impact on cycle time. We have found eight entries to be a sweet spot in balancing IPC and physical design feasibility.

The design has been synthesized using Synopsys Design Compiler 2016.12. The back-end design flow has been carried out using Cadence Innovus-16.10.000. We use the fastest to us available cell library which is an eight track, multi-threshold, multichannel, standard cell library with nominal voltage at 0.8 V from INVECAS. For the cache memories, we use a high-performance, single-port static random-access memory (SRAM) generator provided by INVECAS. The design has been signed off at 902 MHz at 0.72 V, 125°, global worst case (SSG). The floorplan of the chip is depicted in Fig. 3. The final netlist contains 75.34% low-voltage threshold (LVT) and 24.66% super low-voltage threshold (SLVT) cells.

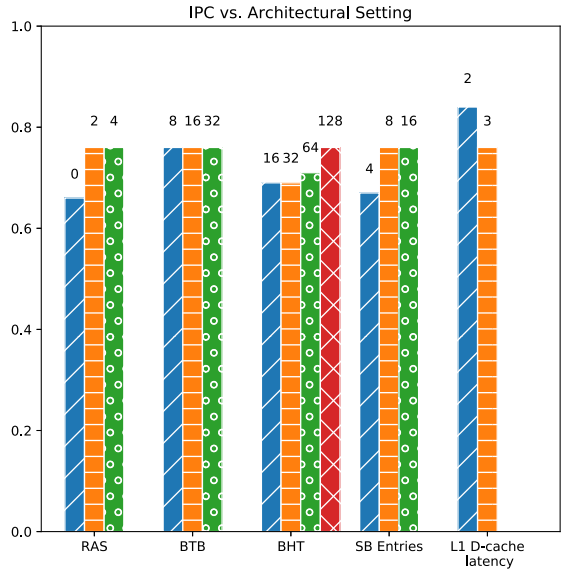

Fig. 2. IPC sweep of different architectural settings from Table III. Each group of bars sweeps one architectural setting while the base configuration is kept stable at RAS: 2, BTB: 16, BHT: 128, ROB entries: 8, L1 D-cache latency: 3. IPC was measured using the Dhrystone benchmark.

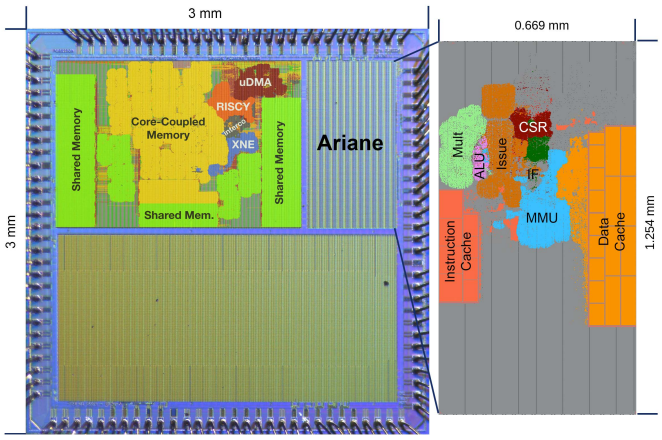

Fig. 3. Ariane has been implemented in GLOBALFOUNDRIES 22 nm on a 3 mm × 3 mm die. The SoC has been separately hardened as a macro (left) and a more detailed floorplan of Ariane (right).

A. Physical Design

To achieve higher clock speeds, several optimizations have been applied during physical design: Useful skew has been used for placement, clock-tree synthesis, and routing to balance the critical paths from and to the memories. Clock shielding was employed to mitigate the effects of crosstalk on the clock tree. Furthermore, we have placed decap cells close to clocktree buffers. Together with a dense power grid, this mitigates the effects of IR drop.

The critical path, with a gate delay of 30 NAND gate equivalents, is around the data caches, as the set-associativity of eight requires significant wiring resources to be routed close to the memory macros. Therefore, dedicated routing channels have been provisioned in the floorplan between the memory macros. Furthermore, due to the high hold times of the memory macros, special attention has been paid to hold-time fixing on those paths by engineering change order (ECO) inserting dedicated buffer cells.

A multimode, multicorner (MMMC)[2] with advanced on chip variation (AOCV) views approach has been used for the entire back-end flow to reduce the margin we need to provision on clock frequency.

IV. RESULTS

We measured our silicon implementation using an ADVANTEST 93000 industry-grade ASIC tester. Postlayout power numbers have been obtained using the postlayout floorplan and netlist of the fabricated design.

A. Methodology

We have developed a number of assembly level tests that exercise particular architectural elements to provide a classification for different instruction groups and hardware modules commonly found in the RISC-V ISA manual. The assembly level tests are synthetic benchmarks, which try to achieve 100% utilization of the respective functional unit under test. To keep code size under control and to hit on a hot instruction cache, such tests contain a (long) sequence of the actual target instructions, which are executed in an infinite loop. The body of the loop containing instructions is dimensioned with 300 instructions, so as to minimize the relative impact of the branch. In particular, we focused on the following

1) *ALU Instructions:* The ALU is used in the majority of RISC-V base instructions. It is used to calculate branch outcomes, arithmetic, and logic operations.
2) *Multiplications:* The multiplier is a fully pipelined two-cycle multiplier, hence it is a relatively big design and consumes considerable amounts of power.
3) *Divisions:* The divide algorithm is a radix-2 iterative divider. Hence, area and power overhead are small but divisions can take up to 64 cycles. An early out mechanism can reduce division time significantly (depending on the operands). The test exercises long divisions.
4) *Load and Stores Without VM Enabled:* For this test, load and stores are triggered subsequently. The cache is warmed-up in this scenario. Address translation is not activated and the program operates in the machine mode.
5) *Load and Stores With VM Enabled:* Similar to the above program except that the program is run in supervisor mode with address translation enabled. Both TLBs are regularly flushed to trigger a page fault and activate the PTW.
6) *Mixed Workload:* This is a generalized matrix–matrix multiplication. This test provides a compute intensive real-world workload triggering all architectural features. Furthermore, this test is used for speed measurements.

The tests have been run on silicon and on the postlayout netlist to provide a detailed per-unit listing. Separate power supplies for the core area, cache memory array, and periphery allow for the detailed power measurements on the actual

[2]Total of 21 corners: functional-mode, worst *RC*, best *RC*, SSG, typical typical (TT), global fast case (FFG), 0.72–0.88 V, −40 °C–125 °C.

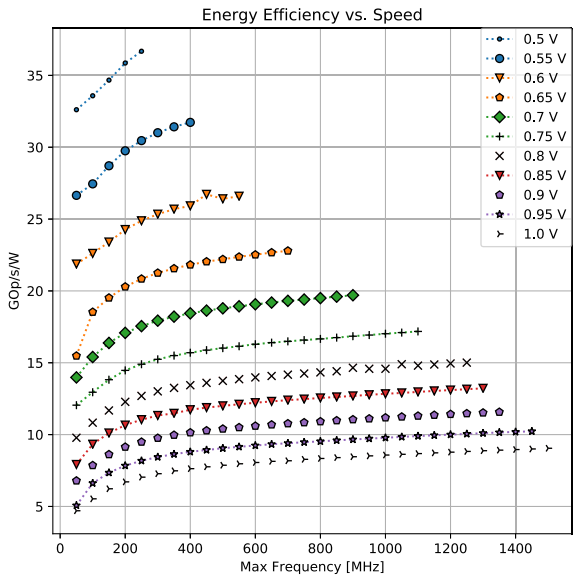

Fig. 4. Detailed power measurement for an integer generalized matrix multiplication (IGEMM) at different operating points (voltage and frequency) and constant bias voltage of 0 V. As the maximum speed is determined by the operating voltage, the most efficient operating point is the maximum achievable frequency at a given VDD. Efficiency decreases for slower frequencies as leakage starts to dominate the power consumption. See Section IV-B for a detailed discussion about power and energy efficiency.

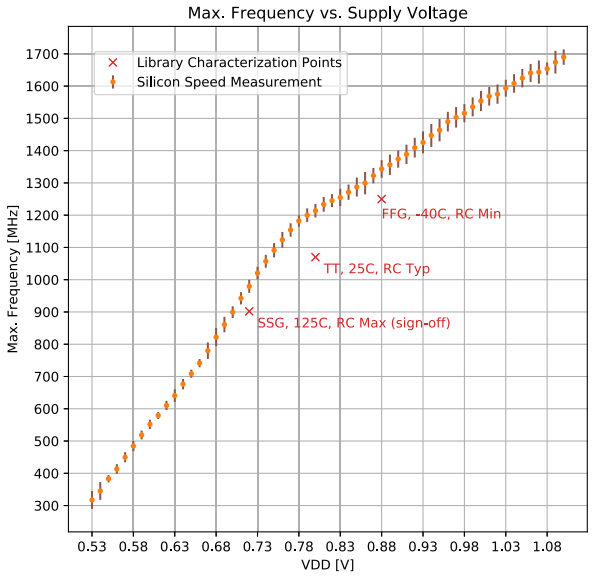

Fig. 5. Average frequency measurement of eiht chip samples under different supply voltages with zero FBB, the error bar indicates the standard deviation of the measured samples. Red: worst, typical, and best case library characterization points. Signoff was done at 902-MHz worst case. Section IV-B1 gives further details on worst case paths and achievable frequency.

silicon. The postlayout and silicon measurements lie within a 10 % error margin, and a calibration factor has been applied to the postlayout power estimates to be aligned with our silicon measurements. We have determined the calibration factor by running a mixed workload benchmark both on silicon and on the postlayout netlist under similar conditions assuming typical case silicon. We provide detailed results in Table IV, separately listing the major contributors to power dissipation.

B. Discussion

We report up to 1.65 DMIPS/MHz for Ariane depending on the branch-prediction configuration and load latency (number of registers after the data cache). On the rather small Dhrystone benchmark, the mispredict rate is 5.77% with a 128-entry BHT and a 64-entry BTB. This results in an IPC of 0.82 for the Dhrystone benchmark.

1) Instruction and Data Caches: As shown in the area overview (Fig. 6) and on the floorplan (Fig. 3), including the size of the SRAM memories (470 kGE data and 210 kGE instruction cache), the largest units are the private L1 instruction and data caches. Furthermore, the critical path in the design is from the memories as the propagation delay of the slower SRAMs adds to the already costly eight-way tag comparison and data selection process. In addition, the wire delay and the diminishing routing resources close to and over the SRAM macros make routing challenging.

Fig. 5 plots the maximum frequency over a large selection of operating voltages. The operating voltage is 0.5–1.15 V with a frequency from 220 MHz to 1.7 GHz (see Fig. 4). Below 0.5 V, the cache SRAMs are no longer functional.

2) Impact of Forward Body Bias: Body biasing (BB) provides an additional tuning knob that allows for trading more

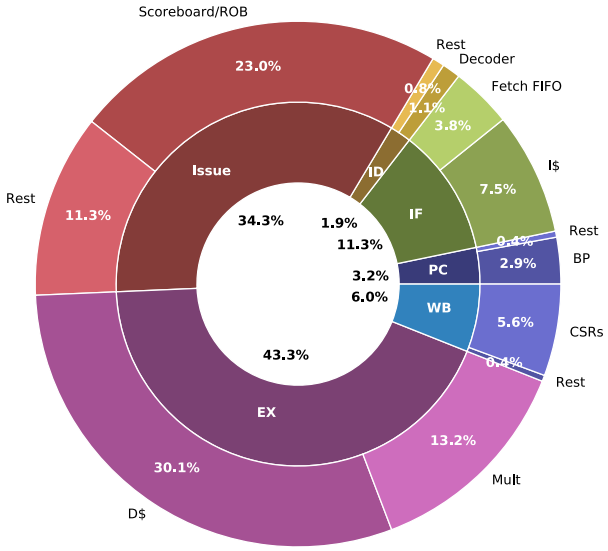

Fig. 6. Detailed area breakdown. The total core area without cache memories amounts to 210 kGE at 1.5 ns.

power for higher frequency. The used standard cells allow for FBB from 0.0 to 1.8 V which results in up to 30% higher speed at 0.5 V. This result confirms that BB is very effective at low supply voltage to compensate for process, voltage, temperature (PVT) variations and to give a significant speed boost for run-to-halt operation. On the other hand, at higher voltages, the achievable speed-up reduces (only 6% at 1 V) and leakage power dissipation increases exponentially.

Furthermore, we have analyzed the impact of FBB on power consumption and energy-efficiency. As FBB will allow us to run the circuit faster, we can reduce the supply voltage to achieve the same speed as without FBB. We can observe the following two effects.

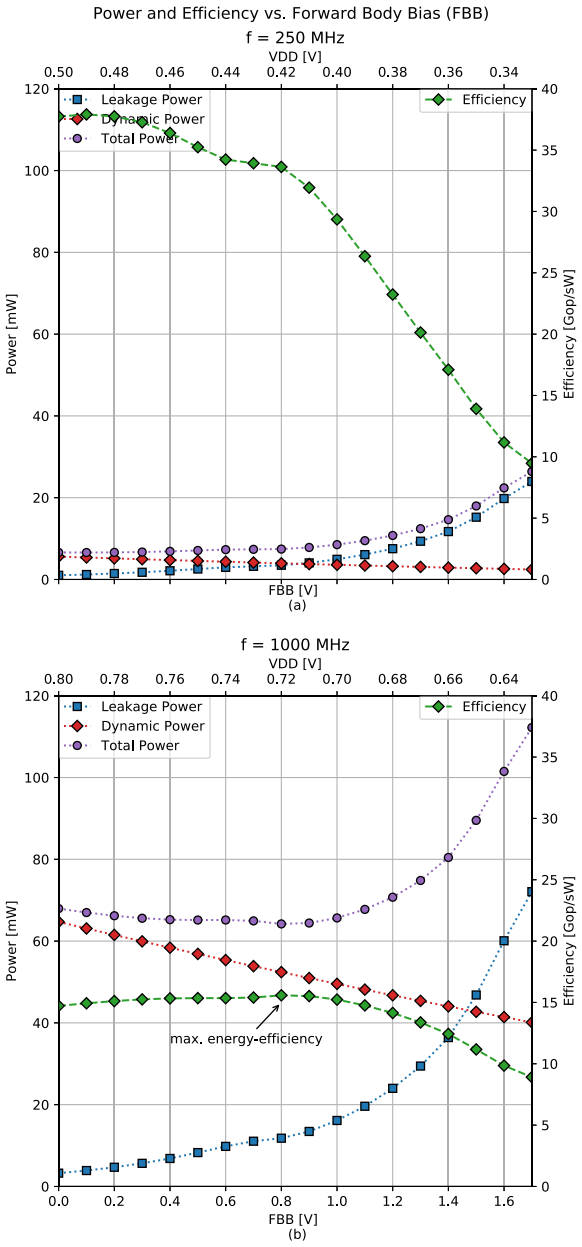

Fig. 7. Leakage power, dynamic power, total power, and efficiency for 0.0–1.7 V FBB and supply voltages at (a) 0.5 and (b) 0.8 V. Dynamic power is decreased as VDD is adjusted to achieve constant frequency under FBB conditions. For 0.1 V FBB, VDD can be reduced by 0.01 V to achieve the same maximum frequency. (a) For low-voltage operation, efficiency decreases with higher FBB as leakage becomes the dominant factor. FBB can be used to trade energy efficiency for performance. (b) For nominal conditions, FBB can be used to increase the energy efficiency.

1) *FBB at Low Voltages*: In this operating regime, leakage power makes up a large part of the total power. Hence, the exponential impact of FBB on leakage dominates the quadratic power savings achieved by reducing, the already low, supply voltage. This impact of this effect on the present work is especially prominent as the circuit was tuned for speed and contains approximately 25% super low threshold cells that have higher impact on leakage (see Section III). Hence, at low voltages, forward body biasing does not improve energy efficiency [as shown in Fig. 7(a)], but it useful as a performance centering method, to ensure that the core can be operated at constant frequency across a wide range of PVT variations.
2) *FBB at Higher Voltages*: At higher operating voltages, the impact of dynamic power on the total power is much higher. Therefore, a decrease in supply voltage reduces the dynamic power which up to 0.8-V FBB outweighs the leakage increase giving an overall energy-efficiency boost of 5%. For more than 0.8 V, the leakage component becomes dominant decreasing the energy efficiency. At this operating regime, FBB can be used to gain energy efficiency [see Fig. 7(b)] as well as for PVT compensation.

3) Application-Class Features: The impact of VM (including TLB and PTW) is significant on the overall power dissipation. In particular, on every instruction and data access, a lookup in the TLB has to be performed which can account for up to 27% of the overall instruction energy while the energy used for actual computation is below 1% in the case of ALU-centric instructions.

Furthermore, the support for user and supervisor modes, and other architectural states such as programmable interrupt vector and status registers, adds a significant area and power overhead on the CSR file. The difference between VM activated and deactivated is quite small as the largest impact on power dissipation is the parallel indexing in the fully set-associative TLB's which is also performed when address translation is disabled. The main difference is the regular page-table walking that does not have a large impact as the PTW is not on the critical path and iteratively walks the tables.

4) Multiplication and Divisions: The multiplier consumes over 13% of the overall core area and can consume up to 4.4 pJ per cycle when active. The serial division unit was optimized for area and energy efficiency, hence its overall impact is minimal.

C. Comparison With Nonapplication-Class Cores

The modular organization of the RISC-V instruction set makes it possible to compare our application-class core to an embedded profile core to get an estimate on the price we pay for the application-class features. The majority of instructions executed is from the base ISA indicate its name in (I) in the RISC-V lingo. Only very few extra instructions and architectural features (related to privilege and VM management) are defined in the privileged specification, which is necessary for the application-class core.

In comparison with a smaller, 32-bit, embedded profile RISC-V core as implemented and reported in [24], we can see a considerable overhead mostly associated with the increased bit width, L1 caches, and the application-class profile of Ariane. Pullini *et al.* [24] report 12.5 pJ in 40-nm technology for an integer matrix multiplication, which is comparable to the workload we used and reported (see Table IV). Adjusting for technology scaling gains from 40 to 22 nm [25], we compare 10 pJ of the small core to 51.8 pJ of Ariane. The same 32-bit core has also been manufactured in GLOBALFOUNDRIES

TABLE IV
ENERGY PER OPERATION CLASS [pJ], LEAKAGE [mW]

Instr. Class	PC	IF Stage		ID Stage		Issue	EX Stage						WB	CSR	CTS	Rest	Tot
		I$	Rest	Dec	Rest		L/S	VM	Mult	ALU	D$	Rest					
Mul	0.30	4.72	0.51	0.01	0.09	1.42	0.22	3.46	0.97	0.02	5.53	0.07	0.05	0.22	4.25	0.76	22.60
%	1.33	20.88	2.26	0.04	0.40	6.28	0.97	15.31	4.29	0.09	24.47	0.31	0.22	0.97	18.81	3.36	100.00
Div	0.25	3.19	0.35	0.00	0.02	1.11	0.22	3.43	0.68	0.00	5.54	0.05	0.02	0.20	4.07	0.81	19.94
%	1.25	16.00	1.76	0.00	0.10	5.57	1.10	17.20	3.41	0.00	27.78	0.25	0.10	1.00	20.41	4.06	100.00
LS w/ VM	0.32	4.63	0.54	0.01	0.09	1.38	0.30	3.50	0.09	0.03	9.18	0.18	0.06	0.22	4.06	0.62	25.21
%	1.27	18.37	2.14	0.04	0.36	5.47	1.19	13.88	0.36	0.12	36.41	0.71	0.24	0.87	16.10	2.46	100.00
LS w/o VM	0.30	4.39	0.51	0.00	0.07	1.36	0.30	3.48	0.07	0.02	9.12	0.17	0.06	0.22	4.04	0.64	24.75
%	1.21	17.74	2.06	0.00	0.28	5.49	1.21	14.06	0.28	0.08	36.85	0.69	0.24	0.89	16.32	2.59	100.00
ALU	0.30	4.36	0.50	0.05	0.13	1.69	0.24	3.47	0.11	0.03	5.53	0.08	0.08	0.24	4.05	0.72	21.58
%	1.39	20.20	2.32	0.23	0.60	7.83	1.11	16.08	0.51	0.14	25.63	0.37	0.37	1.11	18.77	3.34	100.00
IGEMM	0.61	10.17	1.59	0.19	0.65	5.88	0.61	3.84	4.41	0.71	13.75	1.00	0.31	1.12	4.68	2.28	51.80
%	1.18	19.63	3.07	0.37	1.25	11.35	1.18	7.41	8.51	1.37	26.54	1.93	0.60	2.16	9.03	4.40	100.00
Leakage	0.02	0.11	0.02	0.00	0.00	0.12	0.02	0.07	0.08	0.01	0.33	0.04	0.01	0.05	0.00	0.20	1.08

22 FDX on the same die as a part of Ariane's SoC. The small core achieves 12.5 pJ per instruction at 0.8 V [23]. Considering that there is a power overhead involved with the larger SRAMs, standard cell memories (SCMs) and supporting logic, which are also a part of the 22 FDX design results, the estimated energy per instruction is comparable to our technology scaled estimate of 10 pJ.

Most of the overhead stems from the private L1 caches used in Ariane. The memory macros are power hungry and impose significant challenges during physical design. Another contributor to increased power consumption is the larger bit width. The architectural state (register file and CSR file) effectively doubles. This accounts for a 5.7 % (12 kGE) area overhead for the register file and 2.8 % (6 kGE) area overhead for the CSR file. Resulting in larger leakage power and increased switching power, both in the clock tree and on the registers themselves. Furthermore, also the datapath (e.g, the functional units such as ALU and multiplier) suffers from increased complexity, more logic area, and tighter timing requirements resulting in the decreased energy efficiency. Last but not least, the overhead associated with the support for VM is nonnegligible. TLBs and PTW are consuming up to 3.8 pJ per instruction which is a significant 38% of the whole 32-bit core.

Loss of IPC mainly results from mispredicted branches and load data dependencies that need to stall subsequent, dependent instructions because of the three cycles latency of the data cache. Branch prediction can be improved by using more sophisticated prediction schemes such as gshare, loop predictors, or tournament predictors. However, branch prediction is a well-researched topic [26] and has not been explored in this work. The load latency can be decreased at the expense of increased cycle time.

Since the speed of Ariane is higher than the speed reported for the embedded profile core, and its IPC is comparable [6], we conclude that the execution time is lower, although less energy efficient when comparing only the baseline RISCV ISA. However, the ISA extensions proposed in [6] such as postincrementing load and stores, hardware loops, and single instruction multiple data (SIMD) capability show a speedup of up to 10× compared to the baseline ISA. Hence, greatly outperforming pure architectural hardware performance enhancements (such as scoreboarding and branch predicition) both in execution time and energy efficiency.

Table V quantifies the above observation through an example of a compute-bound, 2-D (5×5 filter kernel) convolution of 16-bit data types. The vanilla RISC-V baseline is 129k instructions for the 64-bit ISA and 135k instructions for the 32-bit ISA. When using the DSP ISA extensions, the number of executed instructions drops to 110k instructions. At this stage, all optimizations are automatically inferred by the compiler. Additional hand-tuning and usage of GNU Compiler Collections (GCCs) SIMD builtins of the executed instructions further reduces the instruction count to 29k instructions. While the IPC is higher for the 32-bit core, the higher clock frequency of Ariane significantly reduces the execution time. Nevertheless, all the proposed instruction set extensions reduce the amount of retired instructions by a factor of 4.6 compared to the 32-bit RISC-V baseline, overall resulting in half the execution time compared to Ariane. We can, therefore, conclude that the cost for fundamental "application-class" microarchitectural features is significant, even within a simple, single-issue, in-order microarchitecture. Furthermore, we observe that computer performance and energy efficiency can be boosted more effectively with ISA extensions than with pure clock speed optimization.

D. Comparison to Application-Class Cores

The default Rocket core is a five-stage, in-order core which can be parameterized to be either 64 or 32 bits. The core achieves up to 1.6 GHz in 45 nm [32]. They report 1.72 DMIPS/MHz [33]. The architecture is comparable to

TABLE V
16-bit 2-D (5×5) CONVOLUTION BENCHMARK

	Ariane	RI5CY [6]		
ISA	RV64	RV32	RV32 + DSP	RV32 + SIMD
Instructions [$\times 10^3$]	129	135	110	29
Cycles [$\times 10^3$]	152	137	117	31
IPC	0.85	0.99	0.94	0.93
Freq. [MHz]	1700	690	690	690
Ex. Time [µs]	89.5	198.8	179.7	45.2

Ariane. They achieve a slightly higher IPC of 0.95 at the expense of a slower clock speed. The authors report a core power efficiency of 100 pJ in 45-nm Taiwan Semiconductor Manufacturing Company (TSMC) technology excluding periphery and dynamic random access memory power. Considering technology scaling [25], this would result in 80 pJ per instruction in our target technology which is comparable, but still significantly worse, than 51.8 pJ per operation which we report.

For BOOM, the authors report an IPC of 1.45 for a dual-issue, out-of-order implementation at a frequency of 1.5 GHz in 45-nm TSMC technology. The increase in IPC comes at the expense of a significantly higher hardware complexity of 590 kGE.[3] Kim *et al.* [30] reports 133 pJ per instruction which scales to approximately 100 pJ per instruction in our target technology.

Another application-class core has been developed as a part of the SHAKTI project. The SHAKTI C-class core has been fabricated in Intel 22-nm FinFet technology and consumes about 90 mW and requires about 175 kG logic elements. It also targets midrange computer systems from 200 to 800 MHz. They report 1.68 DMIPS/MHz [28] which translates to an IPC of 0.9. Therefore, the IPC is similar to Ariane but Ariane is running at approximately double the speed of the C-class core and consumes less power at higher speeds. Moreover, no detailed energy breakdown analysis has been published on SHAKTI. Assuming an IPC of 0.9, it requires 122 pJ per operation. Due to the limited amount of information available, it is unclear to the authors what the exact contributors to power are and comparisons might, therefore, be inaccurate.

As remarked earlier, no detailed energy efficiency analysis has been performed on these cores on a functional unit level and we present first results of this kind based on Ariane, our custom RISC-V core, which achieves the best-in-class energy efficiency.

Many studies (see [34] and [35]) have researched the energy efficiency of processors through the high-level design space exploration. However, most of these studies either do not have silicon calibrated analysis or do not go into analysis of the various contributions to energy and cost, mainly because most processors are commercial and have a closed (secret) microarchitecture [36]. Other studies have solely focused on analyzing and improving certain aspects of the microarchitecture like, for example, the register file [37].

[3]Estimated from a similar cell library available to the authors.

Most of these studies on energy-efficient processors were based on proprietary ISA and/or microarchitectures. This is also one of the key novelties of our work on Ariane: not only the ISA is open but also the whole microarchitecture and RTL design. Hence, it is possible to reproduce our results independently, as well as using Ariane as a basis to modify and improve the microarchitecture and its implementation.

V. CONCLUSION

We have presented Ariane, a 64-bit, single-issue, in-order core taped-out in 22-nm FDSOI technology which achieves best-in-class energy efficiency. Based on this microarchitecture, we provide a rigorous efficiency analysis of the RISC-V ISA and its hardware impact.

Furthermore, Ariane has been open-sourced in February 2018 and is available for download on GitHub with a very liberal license for the industry and research community. We provide support for Verilator- and QuestaSim-based RTL simulation as well as a ready-to-go field-programmable gate array (FPGA) bitstream and a prebuilt Linux image.[4]

Our analysis reveals that, although many of Ariane's complex features are necessary to run full-featured OSs, most of the computation can be done on simpler, nonapplication-class cores as they share the same base ISA but lack features such as address translation and different privilege levels. Future work should focus on ISA-heterogeneous systems with microarchitectures consisting of many compute-centric, bare-metal cores, and only a few higher performance application-class management cores, as proposed by early high-level architectural studies [38]. We expect such systems to achieve a high gain in energy efficiency while keeping the programming model reasonable by just using the highly efficient embedded cores for unprivileged compute tasks.

In contrast to licensed ISAs such as advanced RISC machines (ARM) or ×86, the openness of the RISC-V ISA and the availability of encoding space makes it possible to differentiate and explore different architectures and ISA extensions to enhance the efficiency of future computing systems.

ACKNOWLEDGMENT

The authors would like to thank M. Schaffner and F. Schuiki for comments that greatly improved this paper.

REFERENCES

[1] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovi, "The RISC-V instruction set manual: User-level ISA, version 2.0," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2016-118, 2014, vol. 1.
[2] *Greenwaves Gap 8: The IoT Application Processor*. Accessed: Jan. 12, 2019. [Online]. Available: https://greenwaves-technologies.com/en/gap8-product/
[3] *SiFive Core Designer*. Accessed: Jan. 12, 2019. [Online]. Available: https://www.sifive.com/core-designer
[4] *Unleashing Innovation From Core to The Edge*. Accessed: Jan. 12, 2019. [Online]. Available: https://blog.westerndigital.com/unleashing-innovation-core-to-edge/

[4]https://github.com/pulp-platform/ariane/releases

[5] K. Patsidis, D. Konstantinou, C. Nicopoulos, and G. Dimitrakopoulos, "A low-cost synthesizable risc-v dual-issue processor core leveraging the compressed instruction set extension," *Microprocess. Microsyst.*, vol. 61, pp. 1–10, Sep. 2018.
[6] M. Gautschi *et al.*, "Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2700–2713, Oct. 2017.
[7] *Picorv32-a Size-Optimized RISC-V CPU*. Accessed: Jan. 12, 2019. [Online]. Available: https://github.com/cliffordwolf/picorv32
[8] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Amsterdam, The Netherlands: Elsevier, 2011.
[9] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*. San Mateo, CA, USA: Morgan Kaufmann, 2013.
[10] G. Klein *et al.*, "seL4: Formal verification of an OS kernel," in *Proc. ACM SIGOPS 22nd Symp. Operating Syst. Princ.*, 2009, pp. 207–220.
[11] S. Kiamehr, M. Ebrahimi, M. S. Golanbari, and M. B. Tahoori, "Temperature-aware dynamic voltage scaling to improve energy efficiency of near-threshold computing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 7, pp. 2017–2026, Jul. 2017.
[12] I. Hwang and M. Pedram, "A comparative study of the effectiveness of cpu consolidation versus dynamic voltage and frequency scaling in a virtualized multicore server," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, pp. 2103–2116, Jun. 2016.
[13] C. Celio, P.-F. Chiu, B. Nikolic, D. A. Patterson, and K. Asanović, "BOOM v2: An open-source out-of-order RISC-V core," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2017-157, Sep. 2017. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-157.html
[14] K. Asanović *et al.*, "The rocket chip generator," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2016-17, 2016.
[15] N. Gala, A. Menon, R. Bodduna, G. Madhusudan, and V. Kamakoti, "SHAKTI processors: An open-source hardware initiative," in *Proc. 29th Int. Conf. VLSI Design 15th Int. Conf. Embedded Syst. (VLSID)*, 2016, pp. 7–8.
[16] B. Bowhill *et al.*, "The Xeon processor E5-2600 v3: A 22 nm 18-core product family," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 92–104, Jan. 2016.
[17] *EEMBC Coremark Pro Benchmark*. Accessed: Aug. 1, 2019. [Online]. Available: https://www.eembc.org/coremark-pro/
[18] A. Gonzalez, F. Latorre, and G. Magklis, "Processor microarchitecture: An implementation perspective," *Synth. Lectures Comput. Archit.*, vol. 5, no. 1, pp. 1–116, 2010.
[19] A. Bhattacharjee and D. Lustig, "Architectural and operating system support for virtual memory," *Synth. Lectures Comput. Archit.*, vol. 12, no. 5, pp. 1–175, 2017.
[20] G. Tagliavini, S. Mach, D. Rossi, A. Marongiu, and L. Benini, "A transprecision floating-point platform for ultra-low power computing," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2018, pp. 1051–1056.
[21] S. Mach, D. Rossi, G. Tagliavini, A. Marongiu, and L. Benini, "A transprecision floating-point architecture for energy-efficient embedded computing," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.
[22] T. Newsome and M. Wachs, "RISC-V external debug support version 0.13.1," Tech. Rep., 2018. [Online]. Available: https://static.dev.sifive.com/riscv-debug-spec-0.13.b4f1f43.pdf
[23] P. D. Schiavone, D. Rossi, A. Pullini, A. Di Mauro, F. Conti, and L. Benini, "Quentin: An ultra-low-power pulpissimo soc in 22 nm FDX," in *Proc. IEEE SOI-3D-Subthreshold Microelectron. Technol. Unified Conf. (S3S)*, 2018, pp. 1–6.
[24] A. Pullini, D. Rossi, I. Loi, A. Di Mauro, and L. Benini, "Mr. Wolf: A 1 GFLOP/s energy-proportional parallel ultra low power SOC for IoT edge processing," in *Proc. IEEE 44th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2018, pp. 274–277.
[25] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming Moore's law through energy efficient integrated circuits," *Proc. IEEE*, vol. 98, no. 2, pp. 253–266, Feb. 2010.
[26] J. E. Smith, "A study of branch prediction strategies," in *Proc. 8th Annu. Symp. Comput. Archit.*, 1981, pp. 135–148.
[27] K. Asanović, D. A. Patterson, and C. Celio, "The Berkeley out-of-order machine (BOOM): An industry-competitive, synthesizable, parameterized RISC-v processor," Dept. Elect. Eng. Comput. Sci., Univ. California Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2015-167, Jun. 2015.
[28] *Shakti C-Class*. Accessed: Dec. 18, 2018. [Online]. Available: http://shakti.org.in/c-class.html
[29] C. Sun *et al.*, "Single-chip microprocessor that communicates directly using light," *Nature*, vol. 528, no. 7583, p. 534, 2015.
[30] D. Kim *et al.*, "Strober: Fast and accurate sample-based energy simulation for arbitrary RTL," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 128–139.
[31] *Dhrystone Benchmarking for Arm Cortex Processors*. Accessed: May 20, 2019. [Online]. Available: https://static.docs.arm.com/dai0273/a/DAI0273A_dhrystone_benchmarking.pdf
[32] Y. Lee *et al.*, "A 45 nm 1.3 GHz 16.7 double-precision GFLOPS/W RISC-V processor with vector accelerators," in *Proc. 40th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2014, pp. 199–202.
[33] Y. Lee *et al.*, "Raven: A 28 nm RISC-V vector processor with integrated switched-capacitor DC-DC converters and adaptive clocking," in *Proc. IEEE Hot Chips 27 Symp. (HCS)*, Aug. 2015, pp. 1–45.
[34] O. Azizi, A. Mahesri, B. C. Lee, S. J. Patel, and M. Horowitz, "Energy-performance tradeoffs in processor architecture and circuit design: A marginal cost analysis," *ACM SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 26–36, 2010.
[35] Y. Li, B. Lee, D. Brooks, Z. Hu, and K. Skadron, "CMP design space exploration subject to physical constraints," in *Proc. 12th Int. Symp. High-Perform. Comput. Archit.*, 2006, pp. 17–28.
[36] S. M. Tam *et al.*, "SkyLake-SP: A 14 nm 28-core Xeon processor," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 34–36.
[37] X. Zeng *et al.*, "Design and analysis of highly energy/area-efficient multiported register files with read word-line sharing strategy in 65-nm CMOS process," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 7, pp. 1365–1369, Jul. 2015.
[38] X. Liang, M. Nguyen, and H. Che, "Wimpy or brawny cores: A throughput perspective," *J. Parallel Distrib. Comput.*, vol. 73, no. 10, pp. 1351–1361, 2013.

Florian Zaruba (S'18) received the B.Sc. degree from TU Wien, 1040 Vienna, Austria, in 2014, and the M.Sc. degree from the Swiss Federal Institute of Technology Zürich, Zürich, Switzerland, in 2017, where he is currently working toward the Ph.D. degree at the Integrated Systems Laboratory.

His current research interests include design of very large scale integration circuits and high-performance computer architectures.

Luca Benini (F'07) holds the Chair of Digital Circuits and Systems at ETH Zürich, Zürich, Switzerland. He is currently a Full Professor at the Universita di Bologna, Bologna, Italy. He has authored or coauthored more than 900 peer-reviewed papers and five books. His current research interests include energy-efficient computing systems design from embedded to high performance.

Mr. Benini is a fellow of the ACM and a member of the Academia Europaea. He was a recipient of the 2016 IEEE CAS Mac Van Valkenburg Award.