From Swift to Mighty: A Cost-Benefit Analysis of Ibex and CV32E40P Regarding Application Performance, Power and Area

Noam Gallmann
gnoam@student.ethz.ch
ETH Zurich
Zurich, Switzerland

Pirmin Vogel
vogelpi@lowrisc.org
lowRISC C.I.C.
Cambridge, United Kingdom

Pasquale Davide Schiavone
davide@openhwgroup.org
OpenHW Group
Zurich, Switzerland

Luca Benini
lbenini@iis.ee.ethz.ch
ETH Zurich and University of Bologna
Zurich, Switzerland and Bologna, Italy

ABSTRACT

Thanks to the rise of new, not-for-profit, collaborative engineering organizations like lowRISC and OpenHW Group, open source continues to gain traction as an industrially viable computing hardware development paradigm. These organizations continue to develop the Ibex and CV32E40P processors both originating from RI5CY, one of the earliest and most well-known, academic open-source RISC-V designs [1, 2]. Initially, the two cores were clearly differentiated, targeting low-cost control tasks and energy-efficient signal processing, respectively. However, besides establishing industry-grade verification and standards compliance, several performance-oriented features have been added to Ibex, making core and configuration selection less obvious.

This work presents an in-depth comparison of the two cores in terms of application performance, silicon area, and power consumption. While the new Ibex features improve instructions-per-cycle (IPC) performance by up to +34%, they negatively impact the maximum operating frequency. Combining the writeback stage with the single-cycle multiplier yields the highest performance at the highest frequency. This configuration is on par with CV32E40P in terms of performance while consuming 40% less area. CV32E40P has advantages for less-control- but more compute-heavy signal processing tasks where its custom instruction set extensions and optimized pipeline can be put to use.

1 INTRODUCTION

Ibex and CV32E40P, the two 32-bit in-order RISC-V microprocessor cores explored in this work, both originate from a single parent design: RI5CY is among the earliest and most well-known open-source RISC-V processor cores and was originally developed by ETH Zurich and the University of Bologna as main processing element for milliWatt-range edge-computing devices [1, 2]. The extensive use by the industrial community urged the two IPs to be moved to not-for-profit organizations to provide high-quality verification and industrial maintenance, still maintaining their permissive licence and open-source policy.

Ibex (originally known as Zero-riscy) is a 2-stage RV32E,I[M]C core optimized for low-cost and low-power. It has been contributed to lowRISC in December 2018. Since then, Ibex has been extended with new optional features including support for a separate branch and jump target ALU (BT-ALU), an additional writeback pipeline stage (WB-Stage), static branch prediction (SBP), a single-cycle integer multiplication unit (SC-Mult) and the RISC-V draft bit manipulation extension (RV32B).

In contrast, CV32E40P (previously known as RI5CY) is a 4-stage RV32IM[F]C_Xpulp core optimized for high performance and energy efficiency on pattern recognition algorithms. It moved to OpenHW Group in February 2020. The CV32E40P RV32IMC feature set plus interrupts and debug features have been fully verified achieving 100% code-coverage, and the memory interfaces have been modified to be compliant with the OpenBus Interface (OBI) [3]. As a positive outcome of their graduation to industry, both the cores gained substantial effort invested into the design, improving code and design quality, standards compliance, verification, performance and documentation [4, 5].

In this paper, a Power-Performance-Area (PPA) comparison between the two cores is provided. In particular, application performance, silicon area, power and energy efficiency are analyzed across the different RTL parameters and application benchmarks. This paper provides updates on the comparisons of the two cores since the publication of Schiavone et al. based on earlier academic versions of the designs [1].

2 CORE CONFIGURATIONS

All Ibex configurations are built based on GitHub commit *4719edf* [6], and the CV32E40P is built based on GitHub commit *b05bee3* [7]. Detailed documentations of the Ibex an d CV32E40P cores are given in [4] and [5] respectively.

Ibex: The default config of Ibex features a plain 2-stage pipeline as described in [1]. In addition, Ibex offers several options for enhancing performance that we explore in this work: *Writeback Stage* (WB-Stage) for faster memory transactions; *Branch Target ALU* (BT-ALU) to leverage parallel computation of the branch condition and the

CARRV 2021, June 2021, Worldwide

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

branch target, to reduce the latency of mispredicted branches; *Static Branch Predictor* (SBP) which identifies branch and jump instructions and predicts all jumps and conditional backwards branches to be taken and any forward branches to be not taken (not yet fully verified and documented); and *Single-Cycle Integer Multiplication Unit* (SC-Mult), which calculates the *mul* instruction in one cycle and *mulh* in two.

CV32E40P implements the micro-architecture of RI5CY [1, 2]. Although its memory interface changed protocol, performance optimizations are visible only on outstanding transactions busses. Also the timing path from and towards memory has been reduced.

3 EXPERIMENTAL SETUP

Our setup for evaluating performance and power of various core configurations for different benchmark programs and toolchain options is derived from the Ibex Simple System [12]. This instantiates a CPU core, a simulated dual-port RAM memory for instruction and data storage with 1 cycle of latency and wait states, and a basic peripheral to write ASCII output to a file.

3.1 Benchmark Programs

We evaluate the different core configurations using CoreMark [10] and Embench [11] benchmark programs as the first one is a standard benchmark used for embedded processors, and the second one as it provides insights into data-intensive algorithms typical of the edge-computing domain.

CoreMark is a synthetic benchmark program for embedded IoT devices. It implements algorithms such as list processing, matrix multiplications, state machine and cyclic redundancy tests. The outcome of this benchmark is the CoreMark/MHz score.

The Embench suite of benchmarks is a built from a collection of 19 real applications testing the capabilities of CPU cores targeting embedded platforms. The output of this suite is the execution time relative to ARM Cortex-M4 ARMv7 and the code size score using GCC 9.2.0 with optimization level *-O2* and *garbage collection*.

The application binaries are generated using the upstream GCC 10.2.0 RISC-V compiler [8] and the modified 7.1.1 GCC compiler supporting the Xpulp extensions [9].

4 RESULTS

We start by discussing the effects of different toolchain options on code size before analyzing application performance.

4.1 Code Size

Table 1 shows the impact of different compiler toolchain options on code size of Embench, which ranges from 1.13 when code size is optimized as main target, to 6.08 when performance is preferred. The benefits achieved for application performance depends upon the core architecture and is discussed in the following sections.

4.2 Application Performance

We analyze performance results under a range of different aspects.

4.2.1 Compressed Instructions Extension. As shown in Figure 1 (top), the C extension leads to a performance degradation of only 1% on average when running Embench on the default Ibex config,

Table 1: Embench Code Size vs. Toolchain Options

Toolchain Options	Code Size Score
-Os + garbage collection[a]	1.13
-O2 + garbage collection	1.30 (1.15x)
-O3 + garbage collection	1.81 (1.60x)
-O3 + loop unrolling[b]+ garbage collection	2.94 (2.60x)
-O3 + alignment[c]+ loop unrolling	6.08 (5.38x)

a) -ffunction-sections, -fdata-sections, -Wl, -gc-sections
b) -funroll-all-loops, c) -falign-jumps=4, -falign-functions=16

while the code size is reduced significantly by 30% on average, when compiling with *-O2 + garbage collection* (bottom).

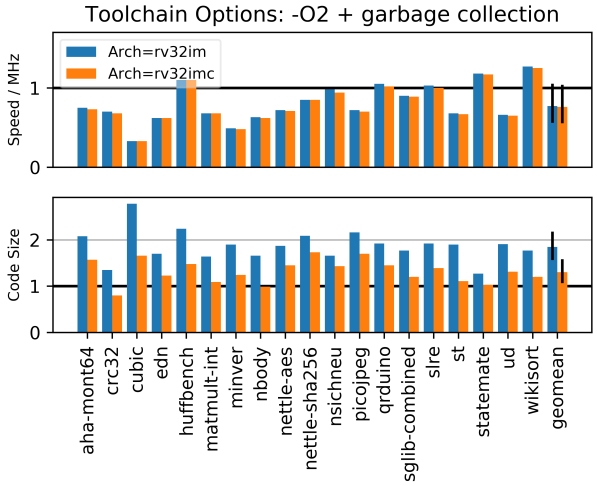

Figure 1: Embench speed (*top*) and size (*bottom*) scores w/ and w/o compressed instructions when using upstream RISC-V GCC and running on the default Ibex config.

4.2.2 Ibex Performance Evaluation. Embench and CoreMark application performance scores for different Ibex configs and toolchain options, including results adjusted for maximum operating frequency, are listed in Table 2. The percentage change figures refer to the default Ibex config with the same toolchain options. Speed scores reported per MHz measure the efficiency of a hardware config in terms of instructions per cycle (IPC). Speed scores reported at maximum frequency take into account the impact of additional complexity on the critical path of a design. A detailed record of the Embench performance for each test program compiled with different compile time optimizations is given in Figure 2.

SBP enhances performance by 1.1% to 3.7% and -0.2% to 6.4% for Embench and CoreMark, respectively. Best benefits are observed for binaries compiled with medium code optimization effort as most jumps and branches are replaced by function inlining and loop manipulations, and loops benefit from the backward-taken, forward-not-taken prediction scheme. However, when maximum frequency is concerned performance decreases by up to 21.9%. The SC-Mult yields consistent performance gains of between 4.3% and 5.7% for Embench, and between 3.1% and 4.6% for CoreMark. Significant improvements are achieved for benchmarks which are heavy on integer multiplications (*edn*, *matmult-int*, *ud*). Since this hardware option does not impact the critical path, the performance gains

translate similarly to the frequency-adjusted scores. The BT-ALU yields performance gains of 2.3% to 5.6% for Embench and 6.2% to 7.8% for CoreMark, which on average are comparable with those of the SC-Mult. Best benefits are observed for binaries which are not heavily optimized for performance as compile-time optimizations often trade an increased code size to reduce control complexity. However, if maximum frequency is targeted, performance boosts are canceled out up to a performance loss of 13.9%. The WB-Stage yields the highest performance gains of the investigated hardware structures (up to 17.2%). It removes the majority of stall cases caused by data transactions, resulting in very high gains for memory intensive workloads. Since this structure does not impact the critical path, the boost in efficiency persists if adjusted for maximum frequency. This is the most area efficient config for CoreMark.

Combining all options yields an increase in IPC performance of up to 34.9% (*All*). The combination of BT-ALU and SBP reduces stalls upon correctly predicted taken branches to 0 cycles if the target instruction is aligned. The combined benefit of the two structures is greater than the sum of their separate speedups, which is observed for the benchmark *matmult-int* for balanced compile-time optimization (Figure 2, middle). The single-cycle multiplier acts orthogonally to those structures, accelerating only integer multiplication instructions. In this config, Ibex performs as fast as the Cortex-M4 using the same compiler optimizations. Exceptions include *cubic*, *minver*, *nbody* and *st* demonstrating less efficient soft-float libraries for RISC-V compared to ARM. Using a more optimized RISC-V soft-float library such as RVfplib [14] can help to reduce this gap. It is worth noting that running a size-optimized binary on the maximum performance Ibex config yields a higher IPC performance than running a performance-optimized binary on the default config. Adjusted for maximum operating frequency dominated by SBP, a performance gain of up to 6.4% remains.

4.2.3 Ibex vs. CV32E40P. When comparing Ibex including WB-Stage and SC-Mult with CV32E40P using the RISC-V RV32IMC ISA, the only differences lie in the multiplication latency for *mulh* instructions and integer division, whereas the highest performing Ibex configs improve the latency of jump and branch instructions. Figure 3 shows a comparison between Ibex and CV32E40P using the balanced compile-time optimization. For benchmarks featuring integer division (*cubic*, *minver*, *nbody*, *nettle-aes*, *sglib-combined*, *st*, *ud*, *wikisort*) a slight advantage is observed for CV32E40P with respect to Ibex with WB-Stage and SC-Mult, as the former implements a variable latency algorithm instead of a fixed 37-cycle latency one. A slightly higher score for Ibex is observed for *aha-mont64*, the only program using the *mulh* instruction. The maximum performance Ibex config yields consistently the highest speed scores, with the exception of *wikisort*, which makes extensive use of integer division.

Table 3 lists average speed scores for Embench and CoreMark when comparing the CV32E40P with the Xpulp extensions enabled and the maximum performance Ibex config. To make a fair comparison the modified PULP toolchain based on GCC 7.1.1 is used for both cores (targeting RV32IM ISA for Ibex and RV32IM_Xpulp ISA for CV32E40P). Compressed instructions have been disabled to exclude performance penalties of the C extension. The average Embench speed score of CV32E40P lies about 11% above Ibex when compiling with general optimization flags.

Table 2: Ibex Application Performance.

Ibex Configuration	Embench Speed / MHz	Embench Speed @ max Freq.	CoreMark Speed / MHz	CoreMark Speed @ max Freq.
Toolchain Options: -Os + garbage collection				
Default	0.71	354	1.58	791
SBP	0.73 (+2.8%)	285 (-19.5%)	1.58 (-0.2%)	618 (-21.9%)
SC-Mult	0.74 (+4.2%)	365 (+2.9%)	1.63 (+3.1%)	805 (+1.8%)
BT-ALU	0.75 (+5.6%)	315 (-11.1%)	1.71 (+7.8%)	718 (-9.3%)
WB-Stage	0.81 (+14.1%)	401 (+13.3%)	1.75 (+10.3%)	867 (+9.6%)
All	0.91 (+28.2%)	358 (+1.1%)	1.96 (+23.7%)	772 (-2.4%)
Toolchain Options: -O2 + garbage collection				
Default	0.76	379	2.07	1034
SBP	0.78 (+2.6%)	304 (-19.7%)	2.20 (+6.4%)	861 (-16.7%)
SC-Mult	0.79 (+3.9%)	389 (+2.7%)	2.15 (+4.0%)	1063 (+2.8%)
BT-ALU	0.79 (+3.9%)	331 (-12.6%)	2.23 (+7.6%)	937 (-9.5%)
WB-Stage	0.88 (+15.8%)	436 (+15.0%)	2.31 (+11.6%)	1147 (+10.9%)
All	1.00 (+31.6%)	393 (+3.8%)	2.79 (+34.9%)	1101 (+6.4%)
Toolchain Options: -O3 + garbage collection				
Default	0.81	404	2.19	1096
SBP	0.84 (+3.7%)	328 (-18.8%)	2.30 (+5.1%)	901 (-17.8%)
SC-Mult	0.85 (+4.9%)	419 (+3.6%)	2.29 (+4.3%)	1128 (+3.0%)
BT-ALU	0.85 (+4.9%)	357 (-11.7%)	2.35 (+7.2%)	988 (-9.8%)
WB-Stage	0.94 (+16.0%)	466 (+15.3%)	2.46 (+12.0%)	1219 (+11.3%)
All	1.07 (+32.1%)	421 (+4.2%)	2.94 (+34.3%)	1160 (+5.9%)
Toolchain Options: -O3 + loop unrolling + garbage collection				
Default	0.86	429	2.35	1173
SBP	0.87 (+1.2%)	340 (-20.8%)	2.43 (+3.6%)	951 (-18.9%)
SC-Mult	0.9 (+4.7%)	443 (+3.4%)	2.46 (+4.6%)	1212 (+3.3%)
BT-ALU	0.88 (+2.3%)	369 (-13.9%)	2.49 (+6.2%)	1048 (-10.7%)
WB-Stage	1.00 (+16.3%)	496 (+15.5%)	2.65 (+13.0%)	1316 (+12.2%)
All	1.12 (+30.2%)	441 (+2.7%)	3.12 (+33.0%)	1230 (+4.9%)
Toolchain Options: -O3 + alignment + loop unrolling				
Default	0.87	434	2.36	1179
SBP	0.88 (+1.1%)	344 (-20.8%)	2.44 (+3.3%)	953 (-19.1%)
SC-Mult	0.92 (+5.7%)	453 (+4.4%)	2.47 (+4.6%)	1218 (+3.3%)
BT-ALU	0.9 (+3.4%)	378 (-13.0%)	2.51 (+6.3%)	1053 (-10.6%)
WB-Stage	1.02 (+17.2%)	506 (+16.5%)	2.67 (+13.1%)	1324 (+12.3%)
All	1.14 (+31.0%)	449 (+3.3%)	3.17 (+34.1%)	1247 (+5.8%)

Table 3: Ibex vs. CV32E40P (Xpulp) App Performance.

Core Configuration	Embench Speed / MHz	Embench Speed @ max Freq.	CoreMark Speed / MHz	CoreMark Speed @ max Freq.
Toolchain Options: -Os + garbage collection				
Ibex + All	0.94	370	1.96	772
CV32E40P	1.05 (+11.7%)	475 (+28.4%)	1.98 (+1.1%)	897 (+16.2%)
Toolchain Options: -O2 + garbage collection				
Ibex + All	1	393	2.79	1101
CV32E40P	1.11 (+11.0%)	502 (+27.6%)	2.74 (-2.1%)	1239 (+12.6%)
Toolchain Options: -O3 + garbage collection				
Ibex + All	1.09	429	2.94	1160
CV32E40P	1.21 (+11.0%)	547 (+27.6%)	3.06 (+4.0%)	1387 (+19.6%)
Toolchain Options: -O3 + loop unrolling + garbage collection				
Ibex + All	1.15	452	3.12	1230
CV32E40P	1.23 (+7.0%)	556 (+22.9%)	3.11 (-0.6%)	1406 (+14.3%)

Using modified PULP toolchain based on GCC 7.1.1 targeting RV32IM for Ibex and RV32IM_Xpulp for CV32E40P.

Figure 4 shows a detailed view on the speed scores of individual Embench programs. The largest performance benefits of CV32E40P are observed for *edn* and *matmult-int* due to the extensive use of hardware loops. Adding in compile-time loop unrolling compensates the lack of support for hardware loops in Ibex and reduces

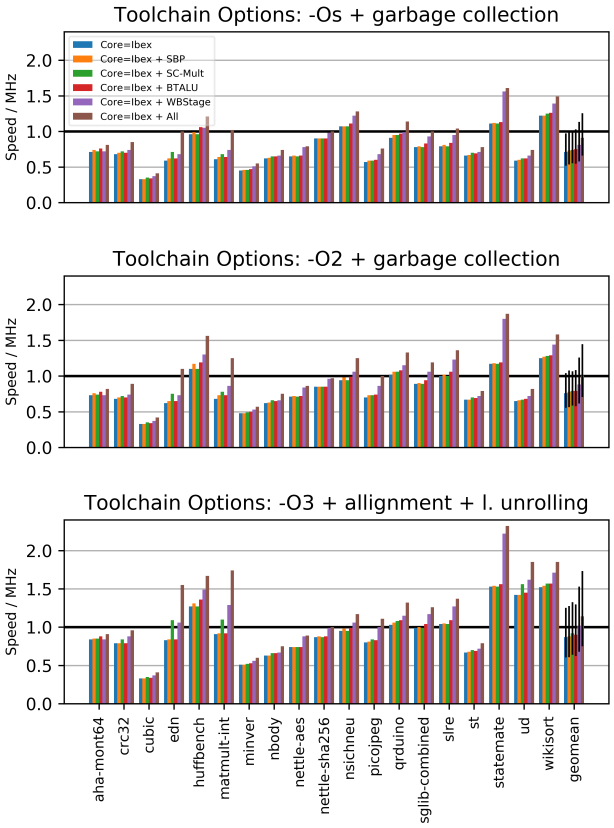

Figure 2: Embench speed scores for different Ibex configurations targeting optimal code size (*top*), balanced code size and performance (*middle*) and maximum performance (*bottom*) referred to the Cortex-M4.

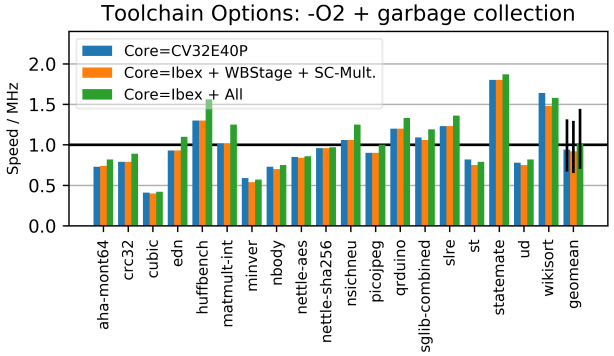

Figure 3: Embench speed scores for CV32E40P and selected Ibex configurations when using the upstream RISC-V GCC toolchain targeting the base RV32IMC ISA.

the performance gap between Ibex and CV32E40P to just 7% (*bottom*). Independently of the toolchain options, consistent advantages for CV32E40P are observed for benchmarks *crc32*, *cubic*, *edn*, *minver*, *nbody*, *nettle-aes*, *nettle-sha256*, *st*, *ud* and *wikisort* thanks to extensive use of the custom Xpulp extensions.

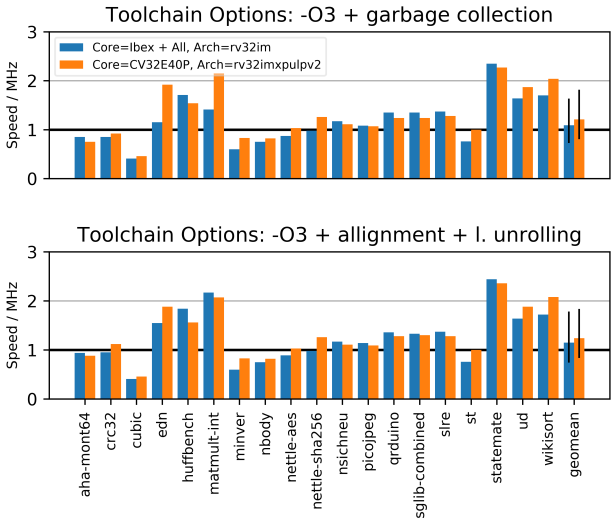

Figure 4: Embench speed scores for Ibex and CV32E40P including Xpulp extensions w/o (*top*) and w/ (*bottom*) loop unrolling (using PULP toolchain without C extension).

Table 4: Core Area and Clock Frequency

Core Config	Area [kGE] @ 100 MHz Total	Delta	@ max Freq. Total	Delta	Max Freq. [MHz]
Ibex Default	23.72	-	31.47	-	500
+ SBP	25.41	+1.69 (+7%)	31.60	+0.13 (+0%)	391 (-22%)
+ SC-Mult	27.40	+3.68 (+15%)	41.87	+10.40 (+33%)	493 (-1%)
+ BT-ALU	24.16	+0.44 (+2%)	29.82	-1.65 (-5%)	420 (-16%)
+ WB-Stage	24.65	+0.92 (+4%)	32.99	+1.52 (+5%)	496 (-1%)
+ WB-Stage + SC-Mult	28.18	+4.46 (+19%)	43.01	+11.54 (+37%)	495 (-1%)
Ibex + All	30.36	+6.64 (+28%)	37.04	+5.57 (+18%)	394 (-21%)
CV32E40P	53.64	+29.92 (+126%)	71.99	+40.52 (+129%)	453 (-9%)

For more control-oriented benchmark programs, such as *aha-mont*, *nsichneu*, *qrduino*, *sglib-combined* and *slre*, the Xpulp extensions have negligible effect [1]. Similarly, the CoreMark/MHz scores of the two cores do not differ significantly as shown in Table 3.

4.3 Core Area and Clock Frequency

The cores have been synthesized in TSMC 65nm technology with typical case conditions (1.2V, 25°) targeting high, low, and regular threshold voltage transistors. The selected synthesis tool was Synopsys Design Compiler 2019.03. A latch-based register file implementation has been used for both the cores. All designs were constrained with a delay of 40% of their respective target clock period on all input and output paths. This is an arbitrary, yet realistic constraint and has an impact on maximum clock frequency. Results may however vary in different design contexts. The maximum operating frequency of each config has been determined by targeting a clock frequency of 500 MHz and then taking the worst negative slack from the synthesis report.

Area and frequency results for the evaluated configurations are listed in Table 4. The percentage change figures refer to the default Ibex config. All config options adding logic through the instruction

memory request, such as SBP and BT-ALU, decrease the maximum frequency significantly (22% and 16% respectively) as the cores have been optimized to trade the cost with the maximum path length towards memories. In contrast, the SC-Mult and WB-Stage options increase only the area cost without impacting the critical path. Enabling all config options reduces the maximum speed by 21%.

The default Ibex config has been designed to minimize area and is as big as 23.72kGE and 31.47kGE when synthesized at relaxed and maximum speed, respectively. The WB-Stage imposes an overhead of 0.92 to 1.52 kGE with relaxed and tight timing constraints, respectively. SBP reduces the maximum operating frequency (from 500MHz to 391MHz) and thus some area savings in the rest of the design can be achieved. This results in no significant area overhead with respect the Ibex default config at 500MHz. Whereas at relaxed timing, an area increase of 1.69kGE is observed. The SC-Mult imposes a significant area overhead of 3.68kGE at 100MHz and of 10.40kGE at maximum speed due to the timing pressure, which drives the synthesizer to select faster multiplier architectures over simpler ones. The area overhead of the BT-ALU amounts to 0.44/1.65 kGE. As the BT-ALU strains the critical path at maximum speed, timing optimization pressure upon other structures is relieved, leading to a smaller overall design. The Ibex config with WB-Stage and SC-Mult enabled is of particular interest. Without notably impacting maximum operating frequency (-1%), this config still enables substantial IPC improvements (2.81 CoreMark/MHz, +19%) thereby achieving a performance level comparable to CV32E40P using the baseline RV32IMC ISA (1390 vs. 1406 CoreMark) despite a 40% lower area footprint. When all the parameters are enabled (All), the area overhead is 6.64kGE (28%) and 5.57kGE (18%) at relaxed and maximum speed, respectively.

When comparing the default Ibex config with CV32E40P, the area of the latter is 126%/129% bigger due to the micro-architecture overhead for the custom ISA extensions and due to the additional pipeline stages. The impact on the critical path is mitigated through separation of decoding and execution in separate stages. Relative to the maximum IPC performance Ibex config, the area overhead is 77%/94% when synthesized with relaxed and tight timing constraints respectively. The maximum operating frequency of Ibex and CV32E40P are at approximately the same levels.

4.4 Power and Energy Efficiency

To estimate the energy efficiency of the different architectures, power simulations have been performed using the post-synthesis netlists of all the core instances. The switching activity has been extracted by means of post-synthesis simulation and analyzed using Synopsys PrimeTime 2019.12. We compare the energy efficiency of the cores when synthesized for 100 MHz and when targeting maximum frequency while executing CoreMark.

The upstream RISC-V GCC toolchain (based on GCC 10.2.0) was used to compile for the RV32IMC ISA of Ibex. For CV32E40P, the modified PULP toolchain (based on GCC 7.1.1) was used to target the RV32IM_Xpulp ISA. In both cases, the *-O3 + loop unrolling* flags have been used for optimization. The dynamic and leakage power as well as the energy dissipated during the benchmark execution is shown in Table 5. Percentage changes are reported with respect to the default Ibex config for a given netlist type.

4.4.1 Power Consumption. Unsurprisingly, the lowest dynamic power consumption of 0.8 mW is observed for the default Ibex config synthesized for an operating frequency of 100 MHz. The SBP increases by 9.1% the dynamic power, and by 15% the leakage. The BT-ALU is the most efficient config option from a power perspective. It imposes the least additional power consumption (5.4% for dynamic and 2.8% for leakage power) that is compensated by an increase in IPC performance of 6.3%. With an increase in leakage power of 7.9%, and an increase of 22.7% and 20.0% in dynamic power, the SC-Mult and WB-Stage options come at similar cost. The dynamic power consumption of the maximum performance config of Ibex is 1.33 mW. This corresponds to an increase of 65.3% with respect to the default config. Leakage power increases by 27%. CV32E40P consumes 60.8% more power through leakage than the maximum performance Ibex config. In terms of dynamic power, CV32E40P consumes just 2.3% more (+69.8% when comparing to the default Ibex config).

When targeting maximum frequency, the default Ibex sets the baseline to 1.09 mW of dynamic and 6.53 μW of leakage power at 500 MHz. The largest overhead in dynamic as well as leakage power is observed for the SC-Mult with 32.3% and 111.3% as the datapath for the multiplication unit is heavily optimized to meet timing constraints. The WB-Stage imposes 21.7% of dynamic power overhead. SBP imposes an overhead of 6.8% in dynamic power while leakage decreases by 1.7%. Similarly, the BT-ALU results in a decrease of dynamic and leakage power by 0.7% and 7.7% due to a reduced maximum operating frequency. However, when normalizing to the frequency (mW/MHz), the SBP dynamic power consumption overhead is the highest (1.36x) when compared with the default config due to a major decrease of maximum frequency. For the maximum performance config of Ibex, dynamic and leakage power increase by 63.3% and 48.9%, while CV32E40P lies at an overhead of 44.2% and 187.9% compared to the default Ibex config. The maximum performance Ibex config consumes 13% more dynamic power than CV32E40P, whereas the leakage power is 49% lower.

4.4.2 Energy Efficiency. The energy consumption is measured as the compound effect of execution time and total power consumption during the execution of the benchmark. When measuring the relaxed netlist energy efficiency at 100 MHz, the only variable impacting the execution time is the IPC. For the default Ibex config the energy consumption is 3.40 μJ. With the BT-ALU, a slight energy reduction of 0.8% can be observed. For the SBP and WB-Stage, we observe moderate dynamic energy overheads (5.6% and 6.1%), while the SC-Mult has the largest dynamic energy overhead (17.3%) due to the little use of multiplications in the selected benchmark compared to the extra power consumption that is burnt at every cycle. The maximum performance Ibex config imposes an energy overhead of 24.3% with respect to the default config. The energy overhead of CV32E40P lies at 29.0%.

At maximum operating frequency, the run time is the compound effect of IPC and maximum frequency. This is particularly noticeable for the SBP config. While the power consumption overhead lies at a modest 6.8%, the increase in overall energy consumption for one CoreMark iteration is almost five times higher (32.1%). The maximum performance config of Ibex consumes 1.44μJ of energy for one CoreMark iteration. That is about 22.2% higher than

Table 5: Core Power and Energy Efficiency Estimations

Core Config	Power		Energy / Iteration		Execution	Freq.	CoreMark	
	Dyn. [mW]	Lkg. [μW]	Dyn. [μJ]	Lkg. [nJ]	Time [ms]	[MHz]	Score / MHz	Score
Netlist @ 100 MHz								
Ibex	0.80	0.18	3.40	0.75	4.24	100	2.36	236
Ibex + SBP	0.88 (+9.1%)	0.20 (+15.2%)	3.59 (+5.6%)	0.84 (+11.5%)	4.10	100	2.44	244
Ibex + SC-Mult	0.98 (+22.7%)	0.19 (+7.9%)	3.99 (+17.3%)	0.78 (+3.1%)	4.05	100	2.47	247
Ibex + BTALU	0.84 (+5.4%)	0.18 (+2.8%)	3.37 (-0.8%)	0.73 (-3.2%)	3.99	100	2.51	251
Ibex + WBStage	0.96 (+20.0%)	0.19 (+7.9%)	3.61 (+6.1%)	0.72 (-4.6%)	3.75	100	2.67	267
Ibex + All	1.33 (+65.3%)	0.23 (+27.0%)	4.22 (+24.2%)	0.72 (-4.6%)	3.18	100	3.17	317
CV32E40P	1.36 (+69.8%)	0.37 (+109.0%)	4.39 (+29.0%)	1.20 (+58.8%)	3.22	100	3.11	311
Netlist @ Max Freq.								
Ibex	1.09	6.53	0.92	5.54	0.85	500	2.36	1179
Ibex + SBP	1.16 (+6.8%)	6.42 (-1.7%)	1.22 (+32.1%)	6.74 (+21.6%)	1.05	391	2.44	953
Ibex + SC-Mult	1.44 (+32.3%)	13.80 (+111.3%)	1.18 (+28.0%)	11.34 (+104.6%)	0.82	493	2.47	1218
Ibex + BTALU	1.08 (-0.7%)	6.03 (-7.7%)	1.03 (+11.1%)	5.72 (+3.3%)	0.95	420	2.51	1053
Ibex + WBStage	1.32 (+21.7%)	7.59 (+16.2%)	1.00 (+8.4%)	5.74 (+3.5%)	0.76	496	2.67	1324
Ibex + All	1.78 (+63.3%)	9.59 (+46.9%)	1.44 (+55.5%)	7.75 (+39.8%)	0.81	393	3.17	1247
CV32E40P	1.57 (+44.2%)	18.80 (+187.9%)	1.12 (+20.9%)	13.37 (+141.3%)	0.71	452	3.11	1406

CV32E40P. Since CoreMark does not make extensive use of the Xpulp ISA extensions, the CV32E40P energy efficiency figures are sub-optimal [1]. The main reason why the CV32E40P energy efficiency is higher than the most performant Ibex config comes from the low-level power optimizations used in the CV32E40P pipeline. In fact, clock-gating is used extensively to silence units like the multiplier to reduce the dynamic power consumption. In contrast, Ibex does not apply any clock-gating (as decoding and execution are merged in one pipeline stage) nor operand silencing via AND-gates.

To reduce the gap, in this work we explore the potential of operand silencing for Ibex. The resulting power figures are reported in Table 6. The indicated percentage changes refer to the corresponding baseline design without this power optimization. Both when targeting relaxed and tight timing constraints, the dynamic power consumption can be reduced substantially with savings between 10% and 35% (SC-Mult at maximum frequency). In addition, the optimization does not significantly impact the critical path.

Table 6: Ibex Power Estimations with Silenced Multiplier

Ibex Config	Power		Freq.	CoreMark	
	Dyn. [mW]	Lkg. [μW]	[MHz]	Score / MHz	Score
Netlist @ 100 MHz					
Default	0.72 (-10.2%)	0.18 (-1.1%)	100	2.36	236
SC-Mult	0.74 (-24.5%)	0.20 (+1.6%)	100	2.47	247
All	1.03 (-22.5%)	0.24 (+5.3%)	100	3.17	317
Netlist @ Max Freq.					
Default	0.92 (-15.6%)	7.01 (+7.4%)	498	2.36	1176
SC-Mult	0.94 (-34.8%)	14.00 (+1.4%)	498	2.47	1231
All	1.39 (-21.8%)	9.87 (+2.9%)	395	3.17	1252

5 CONCLUSION

We have presented an in-depth Power-Performance-Area (PPA) comparison between the closely related, open-source RISC-V processor cores Ibex and CV32E40P including a cost-benefit analysis of various optional performance-enhancement features of Ibex.

We found that the Ibex core architecture is well suited for a versatile range of workloads. Using the optional performance-enhancement features instructions-per-cycle (IPC) performance can be traded for silicon area in multiple steps. Compared to the default 2-stage pipeline configuration, the CoreMark/MHz performance can be increased by up to 34% at an area cost of 28% and 18% for relaxed and tight timing constraints, respectively. However, besides increasing area in particular the BT-ALU and SBP options negatively impact the critical path delay leading to a reduction in maximum operating frequency by 21% and reducing the net performance increase to 6%. Of particular interest is thus the combination of WB-Stage and SC-Mult. This configuration has no notable impact on maximum frequency (-1%) but still enables an IPC improvement of 19% (2.81 CoreMark/MHz). The resulting net performance is on par with CV32E40P using the baseline RV32IMC ISA (1390 vs. 1406 CoreMark) despite a 40% lower area footprint (43 vs. 72 kGE).

For relaxed timing, Ibex can outperform CV32E40P in terms of IPC (3.17 vs. 3.11 CoreMark/MHz) with a 43% lower area footprint (30 vs. 54 kGE). However, achieving this performance level requires compile toolchain options like loop unrolling and alignment that heavily impact code size (+236% compared to -O3). CV32E40P can reach similar performance without inflating code size, thanks to advanced features like hardware loops as well as post-incrementing load/store instructions provided through the Xpulp ISA extensions. More significant advantage is taken of the Xpulp extensions thanks to fused multiply-accumulate and SIMD instructions whenever executing less-control- but more compute-heavy signal processing tasks. For example for Embench, CV32E40P is on average 11% and 28% faster than the maximum performance Ibex configuration in terms of IPC and net performance, respectively. When targeting maximum performance in such workloads, and where the use of non-standard extensions is acceptable, CV32E40P can thus offer higher performance-per-watt efficiency.

As for power, our results show that the default 2-stage pipeline Ibex configuration remains the most efficient option [1]. Enabling the performance enhancements directly translates into an increase of up to 65% and 46% dynamic and leakage power consumption, respectively. The fact that there is room for optimization is not completely unexpected as most effort of lowRISC and OpenHW Group so far has been spent on improving code and design quality, standards compliance, verification, documentation and finally performance. The increase in power is not due to fundamental deficiencies of the performance enhancements and we have shown that there is actually low hanging fruit for improving power consumption and efficiency in Ibex by adapting techniques like input silencing and clock gating that are widely used already in CV32E40P.

REFERENCES

[1] P. D. Schiavone et al., *Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications*, 2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), 2017, pp. 1-8

[2] M. Gautschi et al. *Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices* in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 10, pp. 2700-2713, Oct. 2017

[3] Silicon Labs, Inc., *OBI (Open Bus Interface) 1 Standard*, architecture specification, https://github.com/openhwgroup/core-v-docs/blob/master/cores/obi/OBI-v1.2.pdf

[4] lowRISC C.I.C., *Ibex: An embedded 32 bit RISC-V CPU core*, documentation, [Online, accessed 03-Mai-2021], https://ibex-core.readthedocs.io

[5] OpenHW Group, *OpenHW Group CV32E40P User Manual*, documentation, [Online, accessed 03-Mai-2021], https://cv32e40p.readthedocs.io

[6] lowRISC C.I.C., *Ibex RISC-V Core*, open-source hardware project, https://github.com/lowrisc/ibex

[7] OpenHW Group, *OpenHW Group CORE-V CV32E40P RISC-V IP*, open-source hardware project, https://github.com/openhwgroup/cv32e40p

[8] RISC-V, *RISC-V GNU Compiler Toolchain*, open-source software project, https://github.com/riscv/riscv-gnu-toolchain/tree/58c9d86

[9] PULP Platform, *RISC-V GNU Compiler Toolchain for CV32E40P*, open-source software project, https://github.com/pulp-platform/pulp-riscv-gnu-toolchain/tree/CV32E40P

[10] S. Gal-On and M. Levy, *Exploring CoreMark - A Benchmark Maximizing Simplicity and Efficacy*, The Embedded Microprocessor Benchmark Consortium, [Online, accessed 26-June-2020], https://www.eembc.org/techlit/articles/coremark-whitepaper.pdf

[11] D. Patterson et al., *Embench: An Evolving Benchmark Suite for Embedded IoT Computers from an Academic-Industrial Cooperative - Recruiting for the Long Overdue and Deserved Demise of Dhrystone*, Jun. 2019, [Online, accessed 03-Mai-2021], https://riscv.org//wp-content/uploads/2019/06/9.25-Embench-RISC-V-Workshop-Patterson-v3.pdf

[12] lowRISC C.I.C., *Ibex Simple System*, open-source example hardware system, https://github.com/lowRISC/ibex/tree/master/examples/simple_system,

[13] Embench, *About the Embench results repository*, benchmark results collection, https://github.com/embench/embench-iot-results/tree/8fab201

[14] M. Perotti et al., *RVfplib: A Fast and Compact Open-Source Floating-Point Emulation Library for Tiny RISC-V Processors*, 21st International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS), 2021