Accelerating DSP Workloads on RISC-V Cores and PPA Analysis

Likhith G[1], Kovuru Sai Karthik[2], G P Vishwas[3], G Mallikarjuna Gowda[4], Dr. Poornima G[5]
[1,2,3,4]*Student, Department of Electronics and Communication Engineering, B.M.S. College of Engineering, Bengaluru, India*
[5]*Professor, Department of Electronics and Communication Engineering, B.M.S. College of Engineering, Bengaluru, India*

***Abstract*—Small RISC-V cores in edge devices spend most of their cycles in a few digital signal processing (DSP) loops. These loops can be accelerated inside the pipeline (in-core), by a bus-attached engine (off-core), or by both. Published designs usually report one bundled extension, so the value of each mechanism stays hidden. This paper measures every mechanism separately and in combination on a single system-on-chip. From one five-stage RV32IM core we derive 17 configurations: two baselines, five in-core variants (packed-SIMD multiply–accumulate, post-increment loads, zero-overhead loops, complex arithmetic), five off-core engines (memory-mapped MAC, DMA dot product, scratchpad GEMV, 16-tap streaming FIR, 64-point FFT) and five hybrids. All 17 run four Q15 kernels, a 45-workload sweep and five multi-stage applications (ECG, speech, image, radar, OFDM). Every run is checked bit-exactly against a golden model and against MATLAB double precision. Power, performance and area (PPA) are measured from gate-level simulation and an open-source RTL-to-GDSII flow on SkyWater 130 nm. All 17 layouts are DRC-clean and LVS-matched and meet hold timing in all nine corners. The best hybrid needs 7.48× fewer cycles and 5.0× less energy than RV32IM at 3.5× the area, and its energy–delay product is 37.5× better. After layout the clock is set mainly by repair buffering on long nets, not by the DSP logic.**

***Index Terms*—Design-space exploration, Digital signal processing, Hardware accelerator, Open-source EDA, PPA analysis, RISC-V**

I. INTRODUCTION

Biomedical wearables, hearing aids, cameras, automotive radar and wireless receivers all run fixed-point DSP under a hard time budget and a tight power and area budget. The work is regular: dot products, finite impulse response (FIR) filters, matrix products, recursive (IIR) filters and fast Fourier transforms (FFTs). A scalar processor executes these loops correctly but wastefully. In a typical multiply–accumulate (MAC) loop only one or two instructions in five do arithmetic. The rest are loads, pointer updates, loop counting and branches, and most of the energy goes to instruction fetch and control rather than to the arithmetic [1]. With the end of Dennard scaling, such domain-specific hardware is the main remaining source of performance and energy gains [2].

The open RISC-V instruction-set architecture (ISA) [3] reserves custom opcode space and has a simple memory bus. A designer can therefore extend the core, attach an engine next to it, or do both. The standard packed-SIMD "P" extension is not yet ratified, so custom DSP extensions remain common. Prior work shows impressive individual designs, but each usually bundles many techniques and reports only the combined result. It is hard to answer the questions a designer actually asks: Which mechanism buys the most per unit area? When does an engine beat a better core? What does an idle engine cost? Do pre-layout conclusions survive place-and-route?

This paper answers these questions with a controlled design-space study. Its contributions are:

1) One SoC, 17 configurations. One parameterized five-stage RV32IM core and one bus/DMA interface give two baselines, five in-core, five off-core and five hybrid designs. Only the enabled hardware differs, so every difference in PPA is the cost of one mechanism (Sec. III).

2) Four-level evaluation. The configurations run 4 kernels, a 45-workload parametric sweep (765 RTL runs), a CPU/engine concurrency study (260 runs) and 5 real application chains through a MATLAB/Octave interface. All results are bit-exact against a golden model (Sec. IV).
3) Full PPA. We report cycles, delay, gate-level power, energy and energy–delay product (EDP), together with a cycle identity that explains every count (Sec. V).
4) Sign-off for all 17. Every design goes through the open LibreLane/OpenROAD flow to GDSII on SKY130. We analyse why post-layout frequency drops about 3× (Sec. VII).
5) Design guidelines for choosing in-core, off-core or hybrid acceleration, including memory wait states, multiplier timing and idle-engine clock gating (Sec VIII).

II. BACKGROUND AND RELATED WORK

In-core DSP extensions. The PULP RI5CY core added hardware loops, post-increment addressing and packed SIMD to RV32 for near-threshold IoT nodes [4]. The Snitch core removes load/store and loop overhead in a single-issue core with stream semantic registers and a repetition loop [5]. Recent work adds mixed-precision multi-pumped SIMD to an Ibex core [6], [7] and an exposed-datapath RISC-V DSP in 22 nm [8]. Kang [9] quantifies ISA-extension, datapath-width and pipeline-depth trade-offs stage by stage on an FPGA, and Shinde et al. [10] build a custom RISC-V architecture for real-time ECG feature extraction.

Vector and off-core acceleration. Ara2 is the first open-source processor to support RVV 1.0. It studies lane and core counts on DSP and linear-algebra kernels [11]. Gemmini generates systolic accelerators under a full software stack and shows that host overhead, data movement and command size shape the end-to-end gain [12]. Heterogeneous end-node SoCs such as Marsellus pair a cluster of DSP-extended RISC-V cores with a dedicated 2-to-8-bit DNN engine [13]. Data movement is decisive for such engines; the iDMA engine shows how modular DMA hides memory latency [14]. Platforms such as X-HEEP and the CORE-V eXtension interface standardise how accelerators are attached [15], [16]. A recent survey lists the many ISA extensions and accelerators that now exist [17].

Open physical design. OpenROAD [18] and the LibreLane flow [19], which combines Yosys synthesis with OpenROAD implementation, on the SKY130 PDK [20] make complete open RTL-to-GDSII studies possible.

Gap. Table I compares representative work. Most designs are excellent single points. Few isolate each mechanism under one core, one memory system and one flow, and fewer still carry every point to sign-off. This work fills that gap on a small scale.

TABLE I POSITIONING AGAINST REPRESENTATIVE RECENT WORK

Work	Focus	Acceleration style	Implementation evidence
RI5CY [4]	IoT DSP ISA	in-core (bundled)	silicon
Soft SIMD [6], MaRVIn [7]	mixed-prec. DNN	in-core + HW	ASIC flow
Beaivi [8]	low-power DSP	exposed datapath	22 nm silicon
RV-IM100 [9]	ISA/pipeline DSE	in-core, stepwise	FPGA
ECG core [10]	ECG features	domain core	FPGA
Ara2 [11], Snitch [5]	RVV 1.0 / streaming	vector, SSR	post-layout / silicon
Gemmini [12], iDMA [14]	DNN / data movement	off-core, DMA	VLSI flow / silicon
Marsellus [13]	AI-IoT SoC	DSP cluster + engine	22 nm silicon
X-HEEP [15]	extendible MCU	accelerator platform	silicon
This work	DSP kernels + 5 apps	in-core, off-core and hybrid, each isolated (17 configs)	17 GDSII, SKY130, open flow, DRC/LVS clean

III. SYSTEM ARCHITECTURE

A. The common SoC and core

Fig. 1 shows the system shared by all configurations. The core rv_core is an in-order IF/ID/EX/MEM/WB pipeline. It has full EX/MEM and MEM/WB forwarding, a one-cycle load-use stall and static not-taken prediction, with branches resolved in EX (two bubbles per taken branch). The M extension adds a

single-cycle 33×33 multiplier and a 33-cycle iterative divider. The memories are Harvard, with a 16 KiB instruction ROM and a 16 KiB data RAM. Both are zero-wait-state behavioral models shared by all designs and excluded from area and power. The RAM has one CPU port and a DMA port with two read ports and one write port. Accelerators sit at 0x4000_0000. A simulation-only performance window at 0x5000_0000 is opened by the kernel's first store and closed by its last, so set-up, MMIO programming, polling and post-processing are all counted. The synthesized top level dsp_sys contains the core, the address decode and the accelerator(s). Every option is a Verilog parameter inside a generate block, so a disabled option adds no logic.

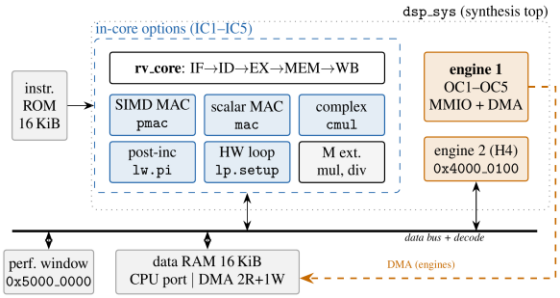


Fig. 1. The SoC shared by all 17 configurations. Blue: optional in-core units (IC1–IC5). Orange: off-core engines with their own DMA port (OC1–OC5); a hybrid (H1–H5) is the IC3 core plus one or two engines. ROM/RAM are excluded from area and power.

B. In-core extensions

Data are Q15 with two samples packed per 32-bit word. A 32-bit accumulator ACC lives inside the core. Table II lists the custom instructions, which use the custom-0 to custom-3 opcodes. One pmac reads one word from each operand and performs two 16×16 MACs. Post-increment loads remove pointer arithmetic but need a second register-file write port. The zero-overhead loop (lp.setup) removes loop counting and branch bubbles. The five in-core variants are IC1 (SIMD), IC2 (scalar MAC + post-increment + loop), IC3 (SIMD + post-increment + loop), IC4 (SIMD + complex) and IC5 (all units). Each one is an exact superset or a minimal step from another, so every step can be attributed to one mechanism.

TABLE II CUSTOM DSP INSTRUCTIONS (Q15, TWO LANES PER WORD; C-N = RISC-V CUSTOM-N OPCODE)

Op.	Instruction	Semantics	Option
c-0	pmac rs1,rs2	ACC += lo·lo + hi·hi (2 multipliers)	SIMD
c-0	mac rs1,rs2	ACC += lo·lo (1 multiplier)	MAC
c-0	accsat rd,sh	rd = sat16(ACC ≫ sh); ACC =0	SIMD/MAC
c-0	cmul rd,rs1,rs2	saturating complex Q15 multiply	CPLX
c-0	padd16/psubh16	lane-wise add/sub (h: halved)	CPLX
c-1	lw.pi / lh.pi	load, then rs1 += imm	POSTINC
c-3	sw.pi / sh.pi	store, then rs1 += imm	POSTINC
c-2	lp.setup rs1,lbl	zero-overhead loop, count in rs1	HWLOOP

C. Off-core engines and hybrids

All engines share one interface: a 64-register MMIO slave, a DMA master on the RAM and a busy flag. The core has no interrupts, so the driver polls a status register. Kernels that an engine does not implement run as plain RV32IM code. The engines differ deliberately in how much work one command does:

- OC1 is a memory-mapped dual-lane MAC peripheral; the CPU moves every operand.
- OC2 is a DMA dot-product engine (one command = one dot product, LEN+1 cycles).
- OC3 is an 8-word scratchpad with a strided 2-D GEMV engine. It computes two output columns in parallel, overlaps write-back, and one command covers a whole matrix product.
- OC4 is a streaming 16-tap FIR: 16 multipliers, an adder tree and a three-stage pipeline produce one output per clock.
- OC5 is a memory-based radix-2 64-point FFT with one butterfly (four multipliers) and a twiddle

ROM. It needs two cycles per butterfly (single write port), i.e., 384 busy cycles per FFT.

A hybrid keeps the IC3 core and adds one engine (H1: FFT, H2: FIR, H3: GEMV, H5: dot) or two (H4: FIR + FFT at 0x4000_0100). Software decides per operation whether the core or the engine runs it. The CPU has its own RAM port, so the core can work while an engine runs.

IV. METHODOLOGY

A. Workloads

The four benchmark kernels are dot64 (64-element Q15 dot product), fir (16 taps × 32 outputs), matmul (8×8) and fft (64-point radix-2 complex). The baseline code is hand-unrolled and load-scheduled, so there are no stalls to remove artificially. The sweep adds 45 workloads (dot 16–1024, FIR 4–64 taps, many matrix shapes, IIR, element-wise and complex operations, FFT). The applications are five chains driven from MATLAB/Octave: ECG QRS detection (Pan–Tompkins, a target of recent RISC-V hardware [10]), speech enhancement, image edge detection, radar pulse compression and an OFDM receiver. Ten single operations complete the set (O1–O10).

B. Verification

An assembler cross-check against GNU as (4002 random words), a golden instruction-set simulator, random-program torture tests and mutation testing (deliberately broken RTL must be caught) secure the core. Randomised descriptors test each engine against an independent model. Every kernel result of every configuration is compared word for word with the golden model, and the gate-level netlist re-runs the kernels with the same result and cycle count. At application level, all 17 versions return identical outputs. These outputs match MATLAB double precision within 16-bit quantisation, and the application results (heart rate, edge map, targets, bit errors) are identical to MATLAB's.

C. Metrics

For configuration c and kernel k with $C_{c,k}$ cycles,

$$S_{c,k} = \frac{C_{B1,k}}{C_{c,k}}, \qquad \bar{S}_c = \left(\prod_{k=1}^{4} S_{c,k}\right)^{1/4}, \qquad (1)$$

where B1 (RV32IM) is the reference. Delay uses each design's own clock, $D=C/f_{max}$. Energy and EDP follow from the gate-level average power $\bar{P}$ at f=100 MHz:

$$E = \bar{P}\, C\, T_{clk}, \qquad EDP = E \cdot D. \qquad (2)$$

Post-layout runs use a relaxed target T=25 ns, so the achievable frequency is

$$f_{max}^{post} = \frac{1}{T - WNS}, \qquad (3)$$

which is below 40 MHz when the worst negative slack (WNS) is negative. Every geometric mean is labelled as cycle (clock-independent) or delay (own clock) where the difference matters.

D. EDA flow

Fig. 2 and Table III summarise the flow. Pre-layout area and timing come from Yosys 0.33 + ABC and OpenSTA on sky130_fd_sc_hd (tt, 25 °C, 1.8 V). Power comes from a gate-level simulation of the running kernel, whose VCD is annotated into OpenSTA report_power. The physical flow is LibreLane 3.1 (OpenROAD floorplan, placement, clock-tree synthesis (CTS) and routing; OpenRCX extraction; multi-corner STA; KLayout DRC; Magic/Netgen LVS) inside the IIC-OSIC-TOOLS container [21]. It uses a 25 ns clock, 40 % core utilisation and 50 % placement density.

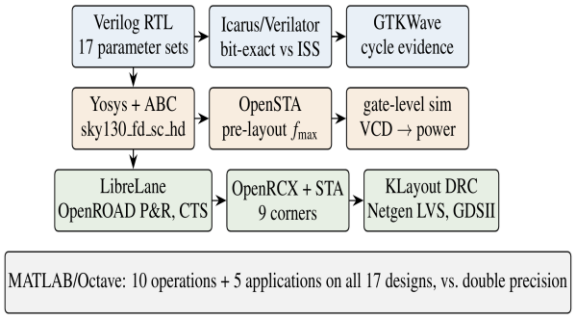


Fig. 2. Verification and PPA flow. Blue: function and cycles. Orange: pre-layout area, timing and power. Green: physical sign-off. Grey: application-level validation.

TABLE III OPEN-SOURCE TOOLS USED

Step	Tool
RTL simulation, lint	Icarus Verilog, Verilator, GTKWave
Synthesis	Yosys 0.33 + ABC
Static timing, power	OpenSTA (VCD-annotated report_power)
RTL-to-GDSII	LibreLane 3.1.0.dev2 [19], OpenROAD [18], OpenRCX
Physical verification	KLayout (DRC), Magic + Netgen (LVS)
PDK, library	SKY130A, sky130_fd_sc_hd [20]
Application layer	MATLAB / GNU Octave

V. RESULTS: PERFORMANCE, POWER AND ENERGY

A. Cycle performance

Fig. 3 gives the cycle speed-up of every configuration on every kernel. The RV32I core without M needs 28× more cycles than RV32IM (geometric mean), because each product becomes a shift-and-add loop. All later comparisons therefore use RV32IM (B1: 318, 1513, 2231 and 6773 cycles) as the reference. The fastest design per kernel is an engine built for that kernel: OC2 on dot64 (49 cycles, 6.5×), OC4 on fir (74, 20.4×), OC3 on matmul (192, 11.6×) and OC5 on fft (395, 17.1×). Each engine stays at 1.0× on the kernels it does not implement. Among the in-core designs, IC5 reaches 2.95× and accelerates all four kernels. The hybrid H4 combines both strengths and reaches 7.48×.

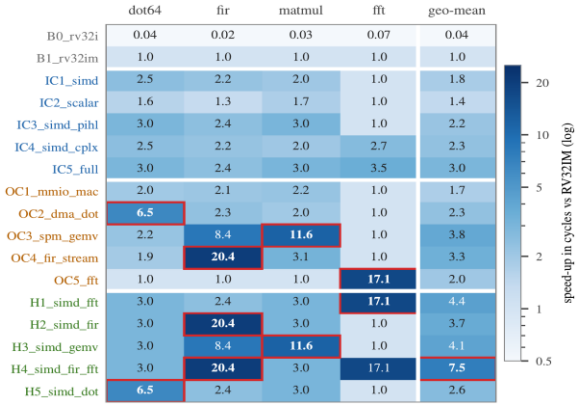


Fig. 3. Cycle speed-up over RV32IM for 17 configurations and 4 kernels (RTL simulation, all runs bit-exact). Red boxes mark the best in each column.

Fig.4 shows these differences directly in the RTL waveforms. The run signal of every configuration falls when its dot64 window closes: 49 cycles for the DMA engine designs, 107 for IC3/IC5 and the hybrids, and 318 for RV32IM.

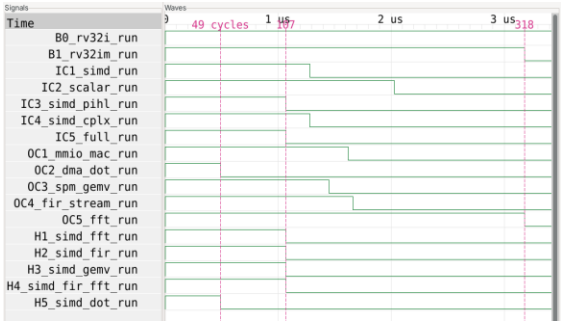


Fig. 4. GTKWave capture of dot64 on all 17 configurations (10 ns cycle). The falling edge of each run signal is the end of the measured window.

Cycle anatomy. Every one of the 68 kernel runs obeys

$$C = N_{instr} + N_{load\text{-}use} + 2N_{taken} + N_{hold} + \varepsilon, \quad |\varepsilon| \le 6, \qquad (4)$$

so each speed-up can be traced to removed instructions, stalls or bubbles. In-core designs run at a cycles per instruction (CPI) of 1.00–1.13, so their speed is set by the instruction count. IC3 executes six instructions per four products against 36 per eight on RV32IM. Its inner loop of four lw.pi and two pmac reaches the single-issue limit of 0.67 MAC/cycle. Engine designs run at CPI 1.5–2.4 on their own kernels. The extra cycles come from polling, where one lw/bnez pair costs five cycles.

Control overhead. Table IV shows the share of cycles in which the engine is not computing. OC5 does a whole FFT per command and wastes only 3 %. OC2 issues one command per dot product and spends 72 % of matmul in driver code and polling, so it reaches only 1.95× there. OC1 shows why an engine must move its own data: with the CPU feeding operands it reaches 1.73×, less than the cheaper IC1.

TABLE IV OFF-CORE ENGINES: WORK PER COMMAND, BEST GAIN AND CONTROL OVERHEAD (SHARE OF CYCLES IN WHICH THE ENGINE IS IDLE)

Engine	One command does	Area	Best gain	Overhead
OC1 MMIO MAC	1 MAC; CPU moves data	1.31×	2.20× matmul	–
OC2 DMA dot	1 dot product	1.24×	6.49× dot64	33–72 %
OC3 SPM GEMV	whole matrix product	1.59×	11.6× matmul	15–48 %
OC4 FIR stream	whole 16-tap FIR	2.75×	20.4× fir	18–49 %
OC5 FFT	whole 64-point FFT	1.43×	17.1× fft	3 %

B. Area and pre-layout timing

The M extension costs +67 % area over RV32I and drops f_{max} from 167 to 100 MHz. Of this, the single-cycle multiplier alone adds 42 799 µm^2 and sets the 10 ns critical path, while the divider adds only 8 632 µm^2. All in-core units are faster than the multiplier stand-alone (pmac 6.0 ns, complex unit 7.0 ns, MAC 5.4 ns). As a result every in-core variant keeps 96–99 MHz at +19 to +52 % area. The smallest overhead is IC1

(+18.8 %) and the largest is OC4 (+175 %, 16 multipliers, about 217 000 µm^2 for the engine alone). Only designs with the FFT engine become slower (OC5 10.38 ns, H1 10.58 ns, H4 10.84 ns). Their butterfly path runs combinationally from the RAM read data to the RAM write data.

C. Power, energy and EDP

Fig. 5 shows the PPA fingerprint of all 17 designs. There is no clock gating in the baseline flow, and 54 % of RV32IM's power is in sequential cells whose clock pins toggle every cycle. Leakage is negligible at 25 ˚C. Power therefore follows the flip-flop count (about 4–8 µW per extra flop at 100 MHz). Average power stays within 8–19 mW, so energy is governed by run time. OC3 uses 3.23× less energy than RV32IM. OC4 is fastest on FIR but gains only 2.31× overall because of its 17.4 mW. H4 is best at 5.01×. In EDP the ranking is H4 (37.5×), H1 (16.0×), H3 (12.5×) and OC3 (12.3×). Fig. 6 shows the area trade-off. The Pareto front runs B1 → IC1 → OC2 → IC5 → OC3 → H1 → H4. OC3 (2.41) and H1 (2.43) give the best speed-up per unit area.

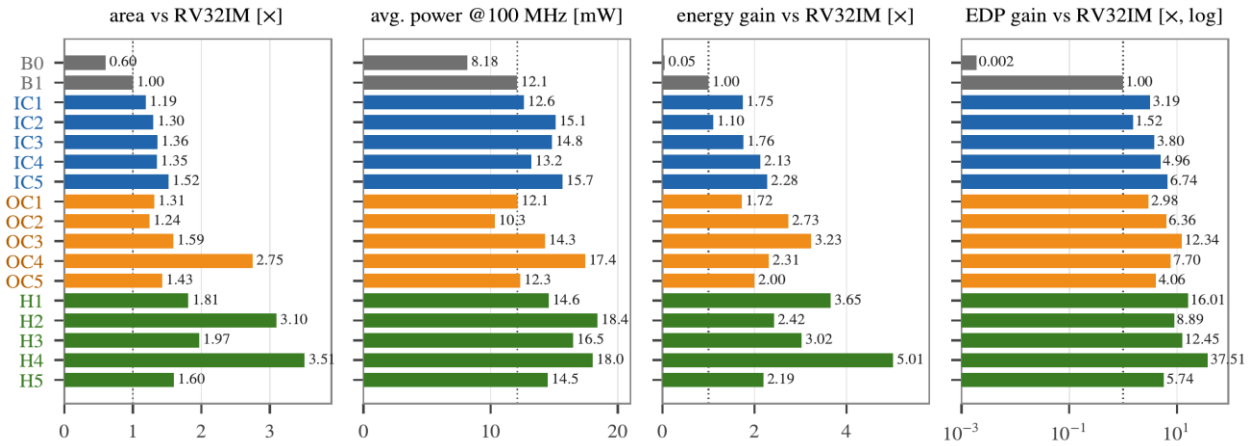


Fig. 5. PPA fingerprint (pre-layout): area, average gate-level power over the four kernels, energy gain and EDP gain against RV32IM (dotted line). Grey: baselines; blue: in-core; orange: off-core; green: hybrid.

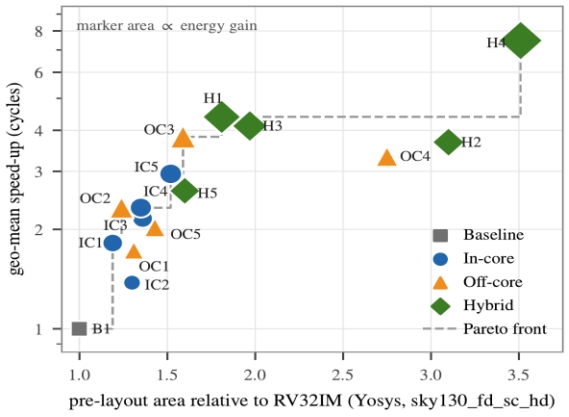


Fig. 6. Area against cycle geo-mean speed-up. Marker area is proportional to energy gain. Designs on the dashed front are not dominated in (area, speed-up). B0 (0.60× area, 0.04×) lies off-scale.

In-core vs. off-core. The in-core group averages 2.06× at a mean area of 1.34×. The off-core group averages 2.53× at 1.66×, but gives no gain in 7 of 20 kernel/configuration pairs (in-core: 3 of 20). Packed SIMD is the best first step (IC1: +19 % area for 1.82×). The scalar MAC variant IC2 is dominated: it is larger than IC1 because of the second write port and the loop unit, yet slower, because it retires half as many products per instruction.

VI. RESULTS: WORKLOADS, APPLICATIONS AND STUDIES

A. 45-workload sweep and ten operations

The winner changes with the workload. OC2 wins dot products at every length (529 cycles for 1024 elements against 4877 on RV32IM). OC4 wins FIR up to 16 taps (14.8 multiplications per cycle) and OC2 wins beyond that. OC3 wins matrix products up to an inner dimension of 16, and OC5 wins the FFT. Only in-core MAC support helps the recursive IIR (1.9×), and IC5's cmul wins complex multiplication. In coverage, IC5 is at least 2× faster than RV32IM on 38 of 45 workloads, ahead of every hybrid (best 35). Fig. 7 shows the delay speed-up of all 17 designs on the ten operations and five applications at their post-layout clocks.

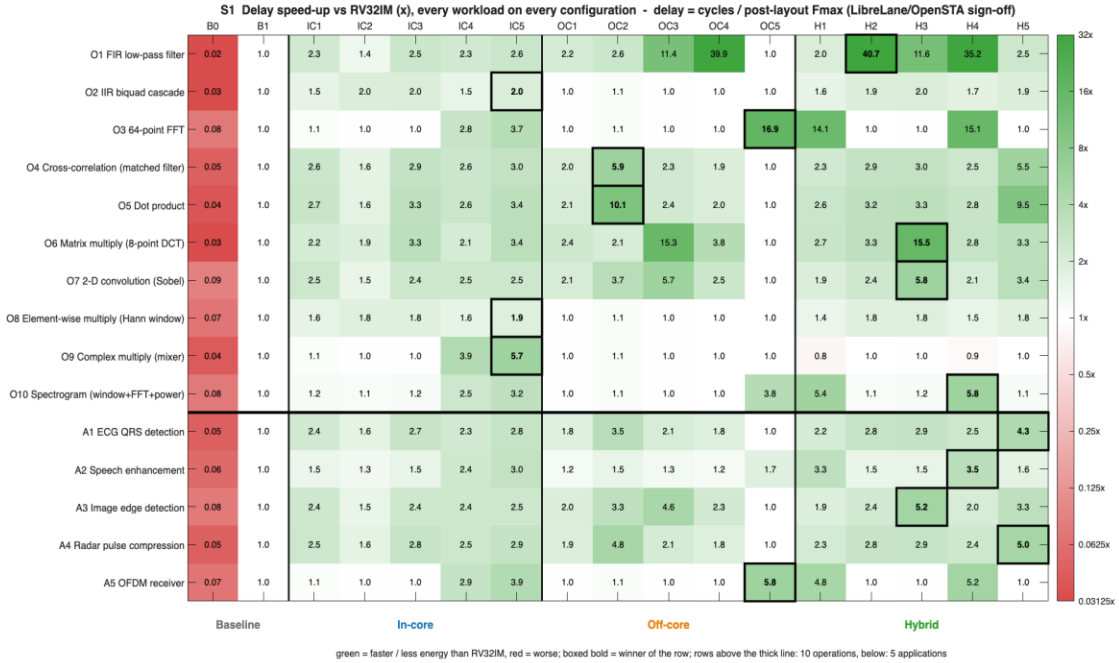


Fig. 7. MATLAB/Octave summary: delay speed-up over RV32IM (cycles divided by post-layout sign-off f_{max}) for 10 operations (top) and 5 applications (bottom) on all 17 configurations. The boxed cell is the winner of each row. All 17 versions return identical results on every row.

TABLE V THE FIVE APPLICATION CHAINS IN DETAIL: INPUT, RISC-V STAGES, BEST DESIGN OF EACH STYLE (DELAY SPEED-UP OVER RV32IM AT POST-LAYOUT SIGN-OFF CLOCKS), LOWEST ENERGY AND APPLICATION RESULT. ALL 17 CONFIGURATIONS RETURN BIT-IDENTICAL OUTPUTS ON EVERY CHAIN

Application	Input signal	RISC-V stage chain	B1 cycles	Best in-core	Best off-core	Best overall	Least energy	Result (RISC-V = MATLAB double)
A1 ECG QRS detection (Pan–Tompkins)	4 s ECG at 360 Hz (76.4 bpm) + baseline wander, 50 Hz hum, noise	IIR high-pass + notch → FIR LP 31 taps → derivative → square → MWI 29 taps	562 391	IC5 2.80×	OC2 3.51×	H5 4.25× (135 767 cyc.)	OC2 3.74×	5 beats, heart rate 76.39 bpm (MATLAB 76.39; true 76.39)
A2 Speech enhancement	0.25 s voiced speech (3 vowels) at 8 kHz + 1 kHz whistle + noise	IIR notch + high-pass → FIR LP 31 taps → 64-pt spectrogram (Hann, hop 32)	947 832	IC5 3.00×	OC5 1.73×	H4 3.52× (237 952 cyc.)	H1 3.31×	SNR −8.36→11.10 dB (MATLAB 11.12 dB); 1 kHz whistle attenuated by 39.6 dB
A3 Image edge detection	32×32 scene (block, disc, ramp, diagonal) + sensor noise	Gauss 3×3 → Sobel G_x → Sobel G_y → G_x^2, G_y^2	344 705	IC5 2.45×	OC3 4.63×	H3 5.22× (69 089 cyc.)	OC3 3.80×	201 edge pixels (MATLAB 201); F1 = 1.00 vs. MATLAB, 0.99 vs. noise-free scene

Application	Input signal	RISC-V stage chain	B1 cycles	Best in-core	Best off-core	Best overall	Least energy	Result (RISC-V = MATLAB double)
A4 Radar pulse compression	two chirp echoes (delays 52, 141) in noise, 256 samples	matched filter (correlation) → echo energy	48 915	IC5 2.91×	OC2 4.80×	H5 4.97× (10 108 cyc.)	OC2 5.11×	2 of 2 targets at delays 52 and 141 (MATLAB and truth identical)
A5 OFDM receiver	64-subcarrier QPSK, 1 training + 2 data symbols, 3-tap multipath + noise	FFT-64 ×3 → [host: 1/H] → cmul equaliser	23 080	IC5 3.89×	OC5 5.75× (3 946 cyc.)	H4 5.16×	OC5 5.74×	0 of 256 bit errors (MATLAB 0); EVM −19.7 dB

B. Application chains

Table V details the five chains. Each chain is a sequence of hardware programs whose inputs are realistic synthetic signals; host-only steps (peak picking, the OFDM channel division, bit decisions) run in MATLAB and are not counted. A hybrid is fastest in four of five chains, and the reason differs per chain.

A1 ECG (Pan–Tompkins). The two long FIR stages (31 and 29 taps) exceed the 16-tap window of the FIR engine, so they map to the DMA dot engine. The IIR filter, derivative and squaring stages need a good core. H5 (IC3 + dot engine) has both and needs 135 767 cycles against 562 391 on RV32IM. OC2 alone is second (3.51×) but uses the least energy (3.74×), because its plain core draws less power. The detected beats and heart rate equal MATLAB's exactly.

A2 Speech enhancement. The recursive notch and high-pass filters can only be accelerated in the core, while the spectrogram is FFT-dominated. A design therefore needs a SIMD core and an FFT engine. H1 and H4 need almost the same cycles (237 888 and 237 952), and H4 wins in delay because its sign-off clock is faster (26.5 vs. 24.7 MHz). H1 has the lowest energy. The pure FFT engine OC5 reaches only 1.73×, because its RV32IM core runs the IIR and FIR stages.

A3 Image edge detection. Every stage is a 2-D convolution or an element-wise square. The scratchpad GEMV engine maps 2-D convolution well, so OC3 reaches 4.63×, and H3 adds the IC3 core for the squaring stages (5.22×, 69 089 cycles). The 201 edge pixels match MATLAB (F1 = 1.00).

A4 Radar pulse compression. The matched filter is one long correlation, i.e., many long dot products, the ideal case for the DMA dot engine. H5 (4.97×) and OC2 (4.80×) are close, and OC2 again has the lowest energy (5.11×). Both targets are found at the true delays.

A5 OFDM receiver. Three 64-point FFTs dominate, so the FFT engine wins (3 946 against 23 080 cycles, 5.75×). The hybrids H1/H4 take the same cycles but run at a slower post-layout clock. IC5 is the best in-core design (3.89×) because of its cmul equaliser. The receiver decodes all 256 bits without error.

Across chains. The stage that the accelerator does not cover dominates the remaining run time (Amdahl's law), which is exactly what a hybrid fixes. H5 was the worst hybrid on the four synthetic kernels yet wins two applications, so synthetic benchmarks alone would have led us to discard it.

C. Core and engine in parallel

The CPU keeps its own RAM port, so a core and an engine can overlap. With a recursive filter on the CPU and an engine job next to it, the measured gain is 93–101 % of the ideal

$$G = \frac{E+C}{\max(E,C)}, \quad (5)$$

with a best case of 1.93×, where E and C are the engine and CPU cycles. For a stream of spectrum frames (window, FFT, power), H1 needs 954 cycles per frame when pipelined (Fig. 8). This is 1.9× better than pipelined OC5 (1822), 3.0× better than IC5 and 9.0× better than RV32IM, and within 4 % of the 915-cycle bound set by the core's own work. The OC5 pipeline is limited by its plain RV32IM core, not by the 395-cycle FFT.

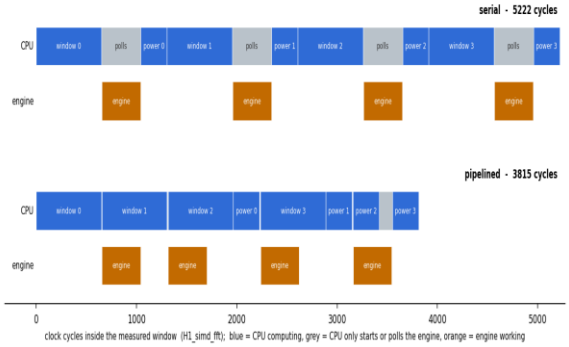


Fig. 8. Measured timeline of four spectrum frames on H1, serial against pipelined (blue: CPU computing; orange: FFT engine; grey: polling).

D. Design studies

Memory wait states. The engines see a zero-wait-state RAM, so their gains are upper bounds. With W wait states per engine beat (Fig. 9), engine cycles grow almost linearly. The FFT engine needs 395, 1550 and 3470 cycles at W=0,3,8. The dot and small-matrix engines fall behind the IC3 core within W=1–3. The FIR and FFT engines, which replace much more work, stay ahead up to W=8.

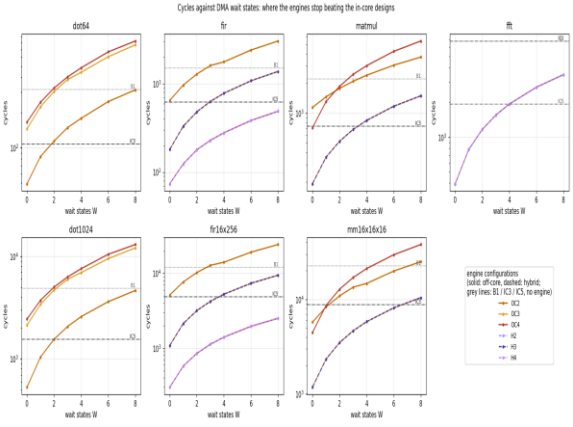


Fig. 9. Cycles against DMA wait states W (539 RTL runs). The horizontal lines are B1, IC3 and IC5. Engines that are only 2–4× better than a SIMD core lose their advantage within one to three wait states.

Two-cycle multiplier. Splitting the 33×33 multiplier into two 33×17 halves raises the clock of RV32IM-based cores by 1.18–1.27×. Wall-clock gains are 1.16–1.23× for the accelerated designs, whose inner loops do not use mul. The change loses for B1 (0.97×) and OC5 (0.96×), whose software is full of multiplies, so it widens every accelerated design's lead.

Idle-engine clock gating. An idle engine is not free: without gating, the unused FIR engine adds +32 % power on the FFT. A gate built from a dlxtn latch and an AND is enabled only on reset, MMIO writes or busy. It leaves all 102 checked RTL runs functionally identical and costs about 1.3 % area. Table VI shows the result; B1 serves as the control and is unchanged.

TABLE VI IDLE-ENGINE CLOCK GATING (GATE-LEVEL POWER, 100 MHZ)

Design	Kernel (engine state)	Before	After	Change
OC4	fft (idle)	14.70 mW	11.20 mW	−23.8 %
H4	fft (idle)	15.80 mW	12.40 mW	−21.5 %
H2	fft (idle)	17.40 mW	13.90 mW	−20.1 %
OC3	fft (idle)	13.80 mW	11.40 mW	−17.4 %
OC1	fft (idle)	13.20 mW	11.70 mW	−11.4 %
OC4	fir (busy)	19.70 mW	18.40 mW	−6.6 %
B1	fft/fir (no engine)	–	–	0.0 %

C intrinsics. Custom instructions issued from C through GNU .insn run 2.5–5.4× faster than plain C. They are still 1.5–2.3× slower than hand-written assembly on FIR, matmul and FFT because of the compiler code around the loops; the dot product reaches assembly speed.

VII. PHYSICAL DESIGN AND SIGN-OFF

All 17 configurations were implemented to GDSII. The 17 runs took 26.0 h in total, 65.6 % of it in detailed routing. Fig. 10 shows the routed layouts and the largest design, H4, in detail. Table VII gives the sign-off metrics read from each run's metrics.json.

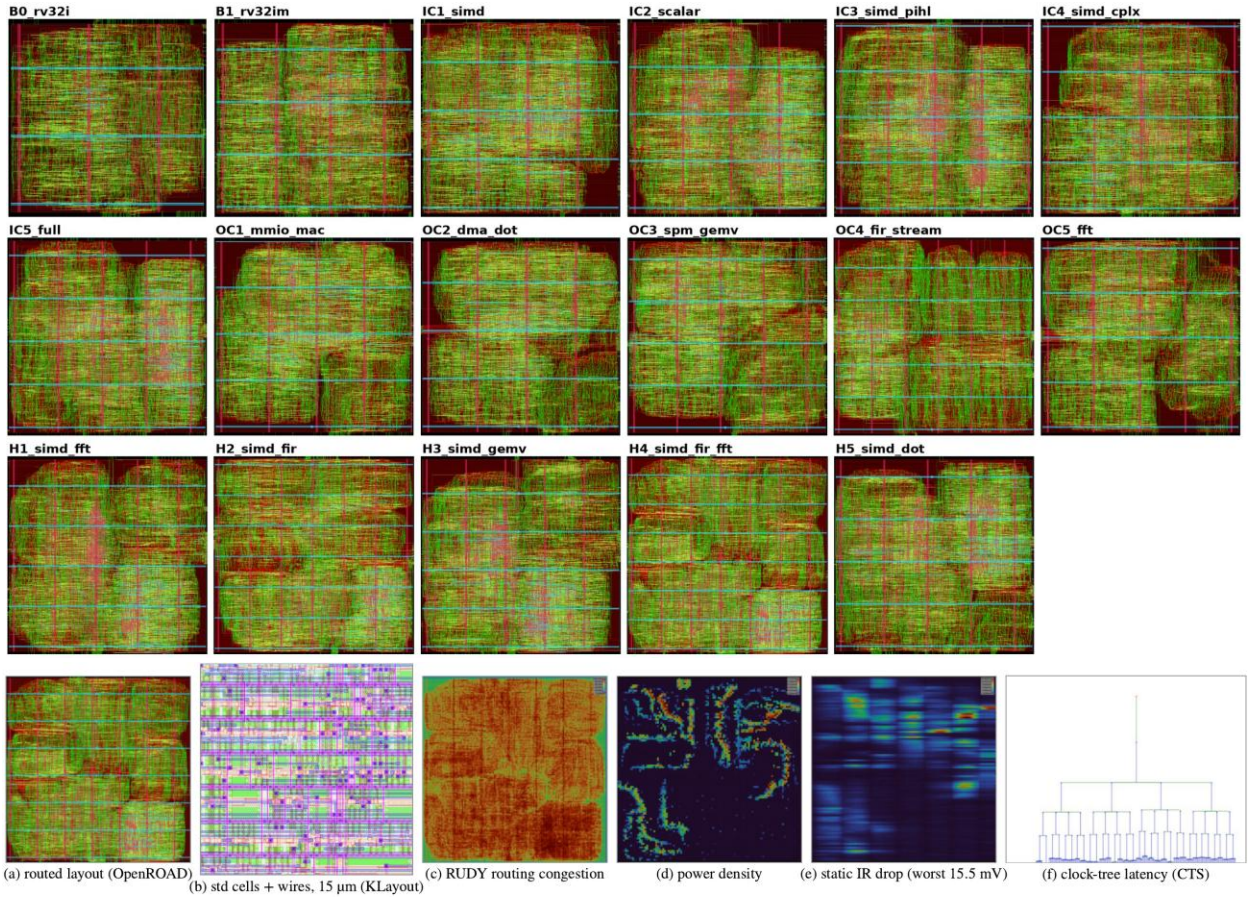


Fig. 10. Physical design. Top: routed layouts of all 17 configurations (OpenROAD, SKY130, sky130_fd_sc_hd, 40 % utilization; each panel is scaled to its own die, which grows from 0.19 mm^2 for B0 to 1.09 mm^2 for H4). Bottom: the largest design, H4 (IC3 core + FIR + FFT engines; 75 001 standard cells, 2.30 m of routed wire), with (a) routed layout, (b) cell-level view, (c) routing congestion, (d) power density, (e) static IR drop and (f) the synthesized clock tree.

TABLE VII POST-LAYOUT SIGN-OFF OF ALL 17 CONFIGURATIONS (LIBRELANE 3.1, SKY130A, 25 NS TARGET). F_{MAX} FROM EQ. (3); DRC (KLAYOUT + ROUTING), LVS AND HOLD ARE CLEAN FOR ALL

	Pre-layout (Yosys/OpenSTA)					Post-layout implementation				Timing (WNS / f_{max})		
Config	Area [µm^2]	×B1	FFs	f_{max} [MHz]	Power [mW]	Die [mm^2]	Std cells	Wire [mm]	IR [mV]	TT [ns]	SS [ns]	SS f_{max} [MHz]
B0_rv32i	76 137	0.60	1395	167	8.18	0.195	10 517	369	0.3	+9.56	−4.27	34.2
B1_rv32im	127 163	1.00	1505	100	12.09	0.328	20 049	713	6.4	+7.19	−8.31	30.0
IC1_simd	151 057	1.19	1544	99	12.61	0.384	24 335	851	19.0	+8.49	−6.53	31.7
IC2_scalar	165 713	1.30	1708	99	15.10	0.410	26 305	972	9.6	+7.91	−7.12	31.1
IC3_simd_pihl	173 535	1.36	1708	99	14.81	0.435	27 597	1039	7.8	+7.73	−7.05	31.2
IC4_simd_cplx	171 060	1.35	1544	97	13.20	0.438	28 282	904	11.3	+8.36	−6.81	31.4
IC5_full	193 701	1.52	1708	96	15.67	0.490	31 746	1120	6.6	+8.65	−6.38	31.9
OC1_mmio_mac†	167 160	1.31	1828	99	12.14	0.420	25 757	890	8.6	+7.75	−7.27	31.0
OC2_dma_dot	157 232	1.24	1599	100	10.34	0.394	24 704	794	7.1	+8.90	−5.40	32.9

	Pre-layout (Yosys/OpenSTA)					Post-layout implementation				Timing (WNS / f_{max})		
Config	Area [µm²]	×B1	FFs	f_{max} [MHz]	Power [mW]	Die [mm²]	Std cells	Wire [mm]	IR [mV]	TT [ns]	SS [ns]	SS f_{max} [MHz]
OC3_spm_gemv†	201 962	1.59	2116	99	14.30	0.512	31 527	1052	18.2	+8.12	−7.38	30.9
OC4_fir_stream	349 657	2.75	2350	98	17.43	0.871	55 511	1536	17.5	+7.51	−8.29	30.0
OC5_fft†	181 792	1.43	1571	96	12.32	0.453	29 407	897	20.2	+3.58	−8.89	29.5
H1_simd_fft†	229 875	1.81	1774	94	14.56	0.562	36 854	1222	25.1	+2.36	−15.43	24.7
H2_simd_fir	394 558	3.10	2553	99	18.41	0.969	64 318	1978	28.3	+7.64	−7.64	30.6
H3_simd_gemv	250 779	1.97	2319	95	16.50	0.617	40 259	1372	53.1	+7.95	−6.85	31.4
H4_simd_fir_fft†	446 847	3.51	2619	92	18.04	1.086	75 001	2298	15.5	+2.23	−12.73	26.5
H5_simd_dot	203 320	1.60	1802	98	14.51	0.507	32 662	1169	5.5	+8.05	−7.44	30.8

†One residual met3 antenna net after detailed routing (OC1, OC3, OC5, H1, H4); all other designs: 0. IR = worst static IR drop. Power = gate-level, VCD-annotated, geo-mean of 4 kernels.

Clean sign-off. All 17 designs show zero KLayout DRC errors, zero routing DRC violations and an LVS result of "circuits match uniquely". Hold slack is positive in all nine corners (worst +0.104 ns). Setup is met at the typical and fast corners: TT slack is +2.23 to +9.56 ns, i.e., 43.9–64.8 MHz. At the slow corner (max_ss, 100 °C, 1.60 V), setup fails by 4.27–15.43 ns, giving 24.7–34.2 MHz. The worst static IR drop is 53.1 mV (2.95 % of 1.8 V, H3); all other designs stay below 30 mV.

Why the clock drops ~3×. Fig. 11 contrasts pre-layout f_{max} (92–167 MHz) with sign-off values. Three causes stand out. (i) The flow ran with area-driven synthesis (AREA 0) and without timing-driven placement. Design repair then inserted clkdlybuf4s50 delay cells as fanout and slew buffers (848–3484 per design). These cells contribute 9–62 % of the worst slow-corner path, e.g., 16.3 of 32.7 ns in B1. (ii) Parasitic extraction alone costs about 3.2 ns at TT in B1 (global-route slack +10.40 ns falls to +7.19 ns at sign-off). (iii) In OC5, H1 and H4, a combinational path from the DMA read data through the butterfly to the DMA write data costs 17–24 % of the clock. This is why H1 drops from second to fifth place in post-layout delay geo-mean (H4 6.61×, H3 4.31×, OC3 3.93×, H2 3.75×, H1 3.61×). For 12 of the 14 designs without the FFT engine, the logic itself accounts for only 2–8.5 ns of the TT critical path. The DSP extensions do not set the post-layout clock. The flow settings and long-net buffering do.

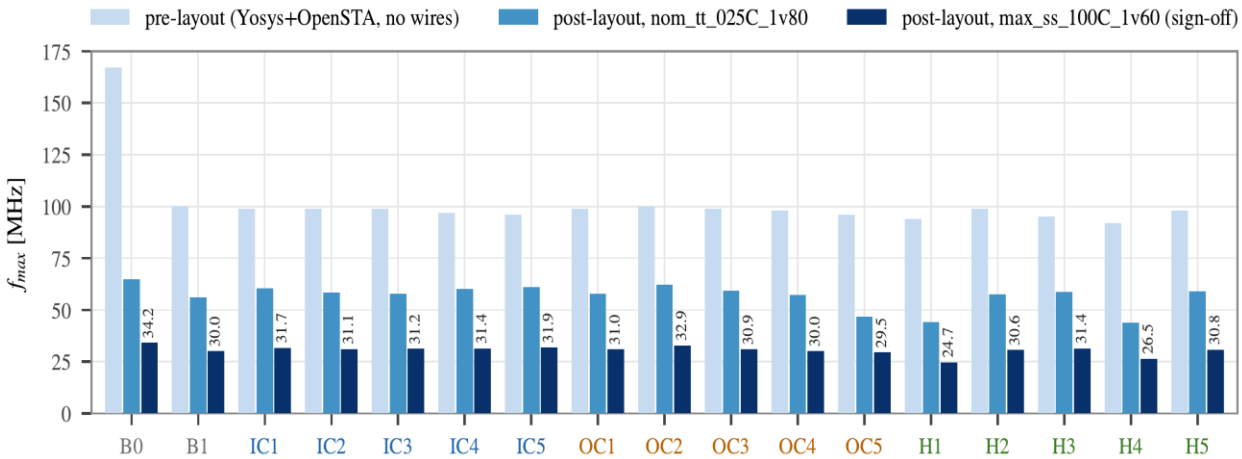


Fig. 11. Maximum frequency before and after layout. Pre-layout synthesis has no wires; post-layout values come from routed, extracted netlists at the typical (TT) and slow (SS, sign-off) corners. The number is the SS value in MHz.

VIII. DISCUSSION AND DESIGN GUIDELINES

Table VIII condenses the study into a per-criterion verdict. Five guidelines follow.

TABLE VIII WHO WINS WHAT (FOUR KERNELS UNLESS NOTED)

Criterion	Winner	Evidence
Cycle speed-up (all)	H4	7.48×, area 3.51×
Best single-technique	OC3	3.82× at 1.59× area
Speed-up per area	H1, OC3	2.43, 2.41
Energy / EDP	H4	5.01× / 37.5×
Lowest power	OC2	10.34 mW, 6.5× on dot64
Coverage (45 workloads)	IC5	≥2× on 38 of 45
Cheapest first step	IC1	+19 % area, 1.82×
Dominated design	IC2	larger than IC1, slower
Post-layout delay (SS)	H4	6.61× at 26.5 MHz
Applications (5)	hybrids	4 of 5 (H5, H4, H3, H5)

1) Start in the core. Packed SIMD with post-increment and hardware loops (IC3) costs no clock and helps almost every loop. It is the right base for any engine.
2) Make engine commands big. A fixed overhead of 9 cycles before and 7 after each busy period cuts the 6.5× dot-product engine to 1.95× on a matrix product that needs 64 small commands. 2-D descriptors (OC3) or whole-transform commands (OC5) avoid this.
3) Engines need fast memory. An engine that is only 2–4× better than a SIMD core loses that margin within 1–3 wait states. Either pipeline its reads or keep its operands in a local scratchpad.
4) Gate idle engines. Clock gating cuts power by 11–24 % while an engine is idle, for about 1.3 % area.
5) Treat pre-layout f_{max} as an upper bound. The ranking by cycles and energy survives layout. The absolute frequency does not, and the FFT butterfly must be registered before an aggressive clock target is attempted.

Limits. Power is from zero-delay gate-level simulation (no glitches or wire capacitance, about ±15 %). The rankings are reliable, but absolute mW values are optimistic. The engine speed-ups assume a zero-wait-state RAM (see the DMA study). Scratchpads are flip-flops rather than SRAM macros. The post-layout runs use a relaxed 25 ns target with area-oriented settings, so sign-off frequencies reflect that flow rather than an optimised timing closure. All signals are synthetic models with realistic noise. Host-only steps (peak picking, the OFDM channel division) are not counted in cycles.

IX. CONCLUSION AND FUTURE WORK

We isolated in-core, off-core and hybrid DSP acceleration on one RISC-V SoC and carried all 17 configurations from bit-exact RTL to DRC/LVS-clean SKY130 layouts. Engines dominate the kernel they were built for (6–20×) and give nothing elsewhere. In-core extensions give 1.4–3× wherever they apply and cost no clock. Hybrids combine the two: H4 achieves 7.48× fewer cycles, 5.0× less energy and 37.5× better EDP than RV32IM, and hybrids win four of five real applications. After layout, frequency is limited by flow-induced buffering rather than by the DSP units. Future work: timing-driven re-implementation with a registered butterfly and a two-cycle multiplier, clock-gated layouts, an IC5-based hybrid (predicted 12.8× on OFDM), the RISC-V architectural test suite, SRAM macros and a small CNN workload for edge-AI use.

ACKNOWLEDGMENT

The authors thank their guide, Dr. Poornima G, Professor, Department of Electronics and Communication Engineering, B.M.S. College of Engineering, Bengaluru, for her guidance throughout this work. They also thank the developers of the open-source tools in Table III.

DECLARATIONS

Conflict of interest: The authors declare that they have no conflict of interest regarding the publication of this paper.

Funding: This work received no external funding.
Ethics statement: This study did not involve human participants or animals. All ECG, speech, image, radar and OFDM signals used in the evaluation are synthetic test signals generated in MATLAB/GNU Octave.
Data availability: The RTL source, test programs, scripts and result tables supporting this study are available from the corresponding author on reasonable request.
Use of generative AI: Generative AI tools (Anthropic Claude) were used to assist with language editing, formatting and the drafting of parts of the manuscript text, figure scripts and documentation. All hardware designs, simulations, synthesis, physical-design runs and measurements were produced with the open-source tools listed in Table III. The authors reviewed, verified and edited all content and take full responsibility for it.

REFERENCES

[1] W. J. Dally, Y. Turakhia, and S. Han, "Domain-specific hardware accelerators," Commun. ACM, vol. 63, no. 7, pp. 48–57, Jul. 2020, doi: 10.1145/3361682.

[2] J. L. Hennessy and D. A. Patterson, "A new golden age for computer architecture," Commun. ACM, vol. 62, no. 2, pp. 48–60, Feb. 2019, doi: 10.1145/3282307.

[3] RISC-V International, The RISC-V Instruction Set Manual, Volume I: Unprivileged Architecture, ISA Specifications ver. 20260120, Jan. 2026. Accessed: Nov. 7, 2026. [Online]. Available: https://docs.riscv.org

[4] 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, doi: 10.1109/TVLSI.2017.2654506.

[5] F. Zaruba, F. Schuiki, T. Hoefler, and L. Benini, "Snitch: A tiny pseudo dual-issue processor for area and energy efficient execution of floating-point intensive workloads," IEEE Trans. Comput., vol. 70, no. 11, pp. 1845–1860, Nov. 2021, doi: 10.1109/TC.2020.3027900.

[6] G. Armeniakos, A. Maras, S. Xydis, and D. Soudris, "Mixed-precision neural networks on RISC-V cores: ISA extensions for multi-pumped soft SIMD operations," in Proc. 43rd IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), New York, NY, USA, Oct. 2024, pp. 1–9, doi: 10.1145/3676536.3676840.

[7] G. Armeniakos, A. Maras, S. Xydis, and D. Soudris, "MaRVIn: A cross-layer mixed-precision RISC-V framework for DNN inference, from ISA extension to hardware acceleration," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 45, no. 5, pp. 2432–2445, May 2026, doi: 10.1109/TCAD.2025.3615505.

[8] K. Hepola et al., "Beaivi: A 22-nm 1-GHz+ exposed datapath RISC-V DSP for low-power applications," in Proc. Design, Autom. Test Eur. Conf. (DATE), Apr. 2026, pp. 1–7, doi: 10.23919/DATE69613.2026.11539538.

[9] H. Kang, "RV-IM100: Quantifying ISA extension, datapath width, and pipeline depth trade-offs in RISC-V microarchitectures," 2026, arXiv:2605.01902.

[10] V. V. Shinde, S. U. Bhandari, D. S. Khurge, S. D. Nagarale, and U. R. Shirode, "A custom reduced instruction set computer-V based architecture for real-time electrocardiogram feature extraction," Int. J. Reconfigurable Embedded Syst., vol. 14, no. 2, pp. 412–427, Jul. 2025, doi: 10.11591/ijres.v14.i2.pp412-427.

[11] M. Perotti, M. Cavalcante, R. Andri, L. Cavigelli, and L. Benini, "Ara2: Exploring single- and multi-core vector processing with an efficient RVV 1.0 compliant open-source processor," IEEE Trans. Comput., vol. 73, no. 7, pp. 1822–1836, Jul. 2024, doi: 10.1109/TC.2024.3388896.

[12] H. Genc et al., "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in Proc. 58th ACM/IEEE Design Autom. Conf. (DAC), San Francisco, CA, USA, Dec. 2021, pp. 769–774, doi: 10.1109/DAC18074.2021.9586216.

[13] F. Conti et al., "Marsellus: A heterogeneous RISC-V AI-IoT end-node SoC with 2-to-8b DNN acceleration and 30%-boost adaptive body biasing," IEEE J. Solid-State Circuits, vol. 59, no. 1, pp. 128–142, Jan. 2024, doi: 10.1109/JSSC.2023.3318301.

[14] T. Benz et al., "A high-performance, energy-efficient modular DMA engine architecture,"

IEEE Trans. Comput., vol. 73, no. 1, pp. 263–277, Jan. 2024, doi: 10.1109/TC.2023.3329930.

[15] P. D. Schiavone et al., "X-HEEP: An open-source, configurable and extendible RISC-V microcontroller," in Proc. 20th ACM Int. Conf. Comput. Frontiers (CF), Bologna, Italy, May 2023, pp. 379–380, doi: 10.1145/3587135.3591431.

[16] OpenHW Group, CORE-V eXtension Interface (CV-X-IF) Specification, 2024. Accessed: Nov. 7, 2026. [Online]. Available: https://github.com/openhwgroup/core-v-xif

[17] S. Keshri, A. Singh, C. K. Palo, S. Adya, and S. Mishra, "RISC-V and machine learning: A survey," J. Supercomput., vol. 82, no. 8, Art. no. 424, May 2026, doi: 10.1007/s11227-026-08463-z.

[18] T. Ajayi et al., "Toward an open-source digital flow: First learnings from the OpenROAD project," in Proc. 56th ACM/IEEE Design Autom. Conf. (DAC), Las Vegas, NV, USA, Jun. 2019, pp. 1–4, doi: 10.1145/3316781.3326334.

[19] FOSSi Foundation, LibreLane: An Infrastructure for Implementing Chip Design Flows, 2025. Accessed: Nov. 7, 2026. [Online]. Available: https://github.com/librelane/librelane

[20] SkyWater Technology and Google, SkyWater SKY130 Open-Source Process Design Kit, 2020. Accessed: Nov. 7, 2026. [Online]. Available: https://github.com/google/skywater-pdk

[21] Institute for Integrated Circuits, Johannes Kepler Univ. Linz, IIC-OSIC-TOOLS: Docker Container for Open-Source IC Design, 2025. Accessed: Nov. 7, 2026. [Online]. Available: https://github.com/iic-jku/IIC-OSIC-TOOLS