

Real-Time MIMO MMSE Equalization Using Standard RISC-V Vector Extensions

Author: Zexu Tan, Sheikh Faizan Qureshi, Emil Matus, Dmitry Utyansky, Pieter van der Wolf, Gerhard Fettweis

Presenter: Zexu Tan

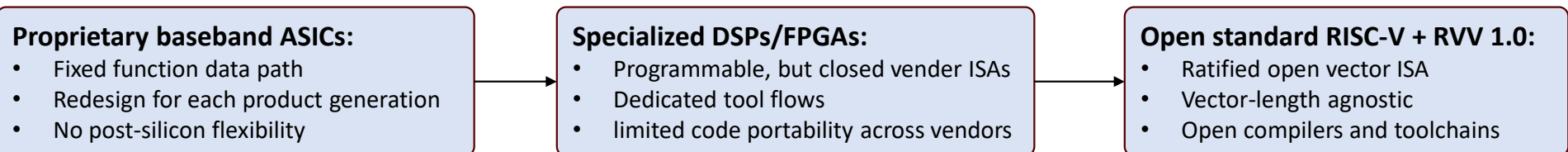
TU Dresden

VODAFONE CHAIR

Problem Statement: Software baseband Needs an Open Vector ISA

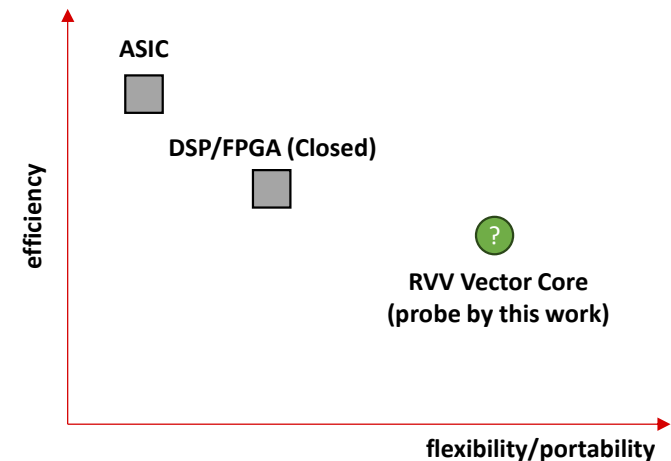
5G New Radio (NR) — wider bandwidth, higher carrier frequency, massive MU-MIMO

→ Physical-layer (PHY) compute demand grows, and industry moves to programmable, virtualized RAN.



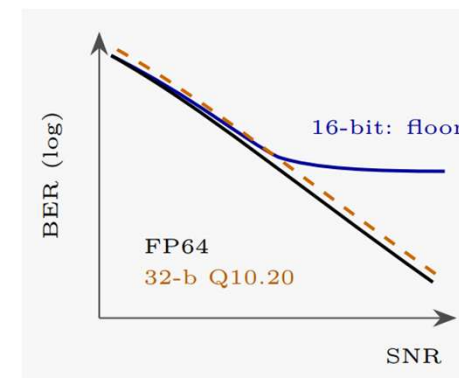
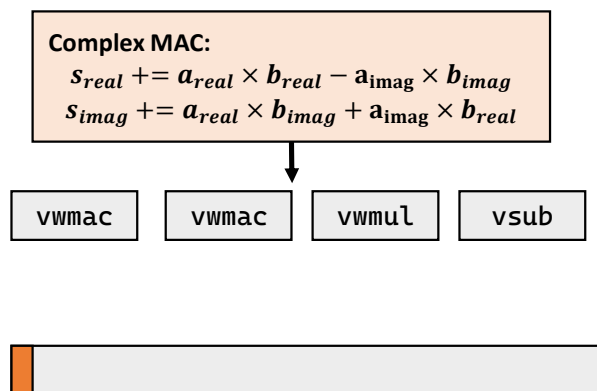
Question: Can standard RVV 1.0 meet the requirements for real-time and error-rate of wireless baseband processing? For which kernel and under what assumptions?

- Case studied in this work — **Fixed-point MIMO MMSE equalizer for OFDM system** (2×2 , 4×4 , 8×8 MIMO), key step in the downlink receiver chain.
- Prior RVV baseband studies cover GFDM and LS/MMSE channel estimation on long-vector cores (Ara).
- DSP capability of RVV so far addressed through non-standard ISA extensions.



Problem Statement : Why MMSE Equalization Stress Standard RVV

- **ISA gap: complex and fixed-point arithmetic**
 - Each complex mac composed from integer widening ops on split I/Q registers
 - No guard-bit accumulators → explicit scaling and saturation at every stage
 - No fixed-point reciprocal square root
- **Workload mismatch: small, serial, matrices in wireless communication**
 - MU-MIMO matrices 2×2, 4×4, 8×8, near-optimal MAC utilization needs $N > 256$ → in-matrix parallelism cannot fill wide vector registers
- **Precision vs. Error Rate**
 - Accumulated quantization error in 16-bit fixed point results in above 25 dB SNR
 - Near-optimal output SNR performance requires 32-bit data format, which half the throughput, double the data traffic
- **Microarchitectural Factors**
 - 32 architectural registers (LMUL=1): spills/refills degrade cycle performance → explicit register management
 - In-order cores, no compiler software pipelining → manual instruction rescheduling
 - VLSU width and chaining dominate: 2.5× between two RVV cores, same kernel and config



Quantization error in 16-bit data results in a performance floor

Outline

- Problem Statement
- Introduction: Why RISC-V Vector (RVV) for DSP Extension and wireless communication?
- Methodologies: Mapping the MIMO Equalizer Kernel to RISC-V Vector Processor
- Evaluation: RTL Cycle Performance of RVV platforms and Real-Time Requirements of 3GPP standards

Why RISC-V Vector (RVV) for DSP Extension?

Packed SIMD:

X86 SSE/AVX, Arm NEON

- Vector length hard-coded in the ISA
- Maintain high instruction throughput based on superscalar issue, out-of-order execution, register renaming and etc.

VLIW SIMD DSPs:

TI C6x, CEVA-XC, Hexagon

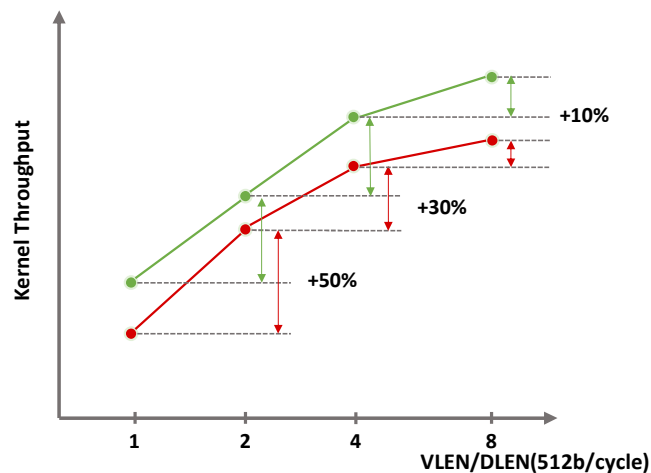
- DSP efficient, but static software scheduling
- Vendor closed ISA and tool flows, almost zero portability
- Code retuned for every pipeline revision.

RISC-V Vector 1.0:

open standard, ratified 2021

- Vector-length agnostic
- Chaining and per-element sequencing are hardware's job
- SEW / LMUL / masks are configured at software run time.

The efficiency metric: Chime Length of a vector instruction **VLEN/DLEN**



One instruction executes **VLEN/DLEN beats**, instruction fetch, decode and issue overhead are amortized.

Vector instruction chaining and decoupled memory access instead of out of order Execution.

Deep folding reaches DSP-class efficiency from a simple in-order vector core, no VLIW scheduling, no width-locked rewrites

→Find the design point: VLEN and DLEN are microarchitecture knobs, while the chime length determines overhead and throughput.

RISC-V DSP Task Group:

- **RVV Digital Signal Processing Task Group (RVV DSP TG):** [RVV Digital Signal Processing Task Group \(RVV DSP TG\) - Vector SIG - RISC-V Tech Hub](#)
 - Enhanced Fixed Fixed-point Arithmetic Instructions
 - Complex Arithmetic Instructions
 - Instructions defining unified interfaces to DSP-specific hardware acceleration blocks.
 - Reduced ISA of RVV
 - ...
- **RISC-V vector for wireless extension (xcmvw):** [gaoshanlee193/riscv-vector4wireless-extension: RISC-V vector extension for wireless signal processing base on RVV1.0](#)
 - Proposed by **Xinsheng Technology Co. LTD.**, a strategic member and proposer of the RISC-V DSP Task Group
 - Targeted for the wireless communication signal processing
 - Extended on fixed-point complex arithmetic, format conversion, dynamic scaling multiply-accumulate instructions, and etc.
 - Add extra micro-architecture and CSRs to mimic traditional DSPs.

An Overview on the RTL Simulation Platforms

- **Saturn: A RISC-V Vector Processor with Monolithic VRF model and SIMD Datapath**
 - Open-source IP developed by UC Berkeley, part of the CHIPYARD framework.
 - Coupled with RISC-V scalar core, Rocket-Chip
 - Features short vector efficiency, supporting efficient dynamic scheduling of short-chime short-vectors.
 - Adopted hardware configuration: VLEN=4096 bit, DLEN=512b bit/cycle.
- **ARA2: A RISC-V Vector Processor with Banked VRF model and split-lane DLP**
 - Open-source IP developed ETH Zürich, part of the PULP-platform.
 - Coupled with RISC-V scalar core, CVA6
 - Features long vector scalability in HPC, implemented with split-lane VRF and reduced memory bandwidth
 - Adopted hardware configuration: VLEN=4096 bit, DLEN=512b bit/cycle.

Vectorizing the MIMO MMSE Equalizer Kernel

Kernel vectorization with 3GPP Resource Grid:

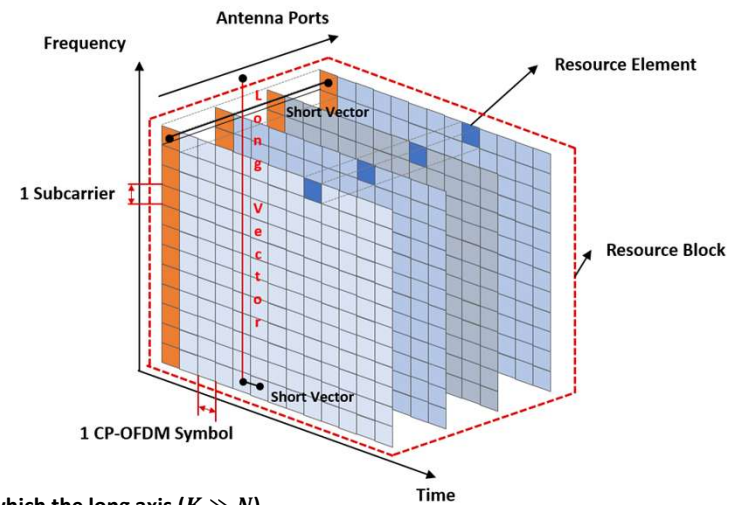
- MIMO dimension N (2×2 , 4×4 , 8×8) in an OFDM receiver system limits matrix size in the equalizer kernel.
- Near-optimal efficiency for in-matrix parallelism requires $N \geq 256$

A better solution is to vectorize across subcarriers K

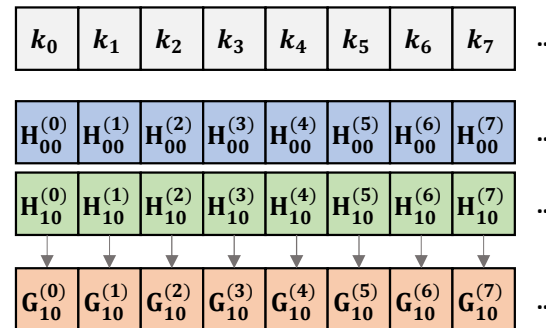
- $vl = VLEN/SEW = 4096/32$ (this work) = 128
- Vector register is fully utilized at any MIMO dimension
- Matrix/vector multiplication as outer product

Properties and cost:

- Vector-length agnostic: strip-mining over K , the same kernel for any VLEN.
- Each kernel invocation estimates $N \times vl$ transmitted symbols

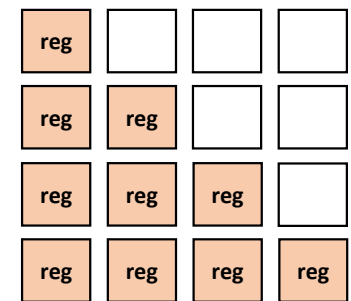


K subcarriers, which the long axis ($K \gg N$),
for each vectorized MMSE instance:



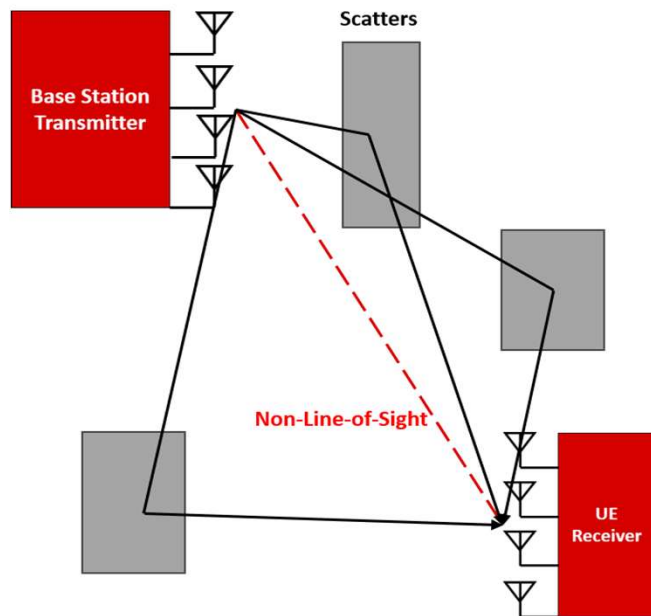
$G_{10} += H_{00}^* H_{10}$ for all k subcarriers at one batch

4×4 MIMO across vector registers:



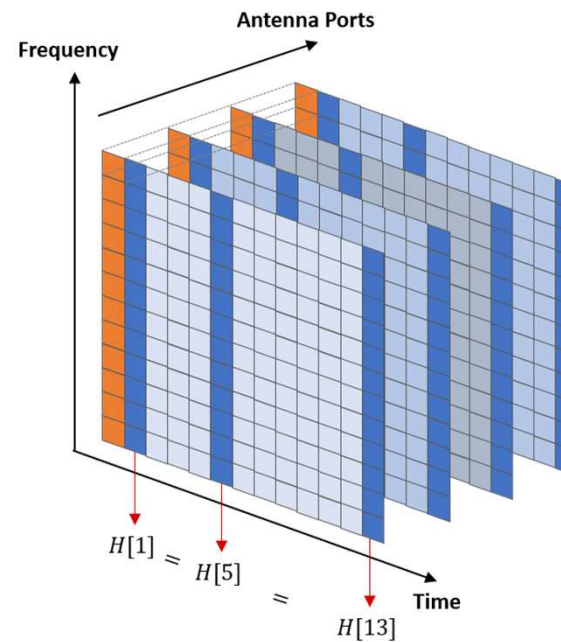
Lower triangular part computed due to Hermitian

Rayleigh-Fading and Block-fading Channel Model



Rayleigh-Fading Channel:

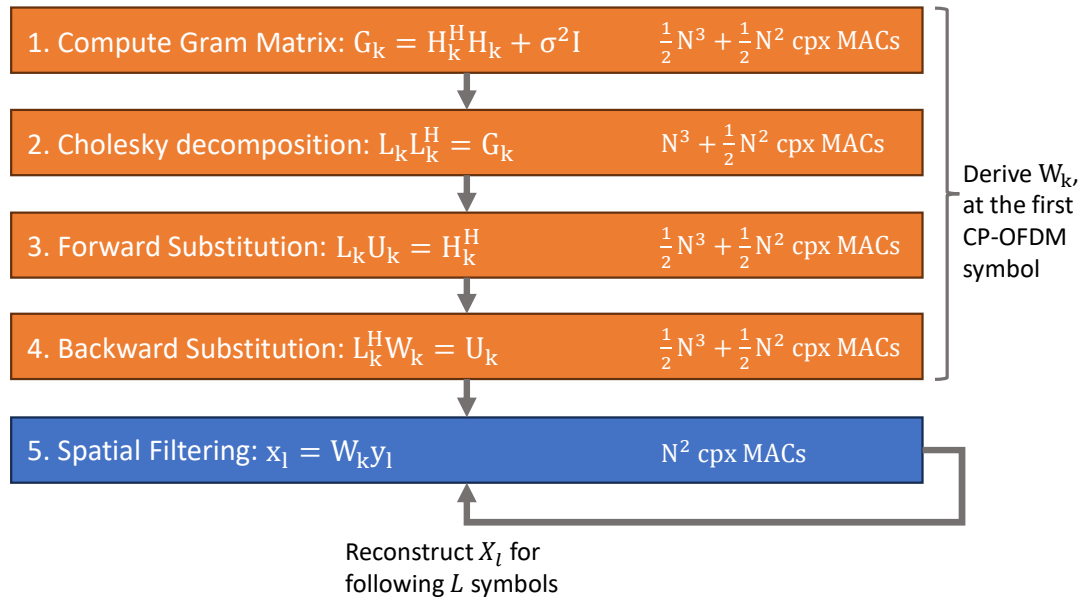
- Signals have many scattered path, and no line of sight
- Multi-path delay spread causes frequency selectivity



Block-Fading Channel:

- Low speed UE devices, ≤ 30 km/h $\rightarrow$ low Doppler
- Equalizer weightc matrix W is shared across the time block

Algorithm Steps of the MIMO MMSE Equalizer Implementation



Some practical configurations from 3GPP 5G NR

- LTE-High End, $K=1200$, $L=120$
- 5G FR1, $K=3200$, $L=120$
- 5G FR2, $K=790$, $L=100$

System Model: $y = Hx + n$, $H \in \mathbb{C}^{N \times N}$, $y, x, n \in \mathbb{C}^{N \times 1}$

MIMO MMSE equalizer: $W = (H^H H + \sigma^2 I)^{-1} H^H$

- Matrix inversion performed by Cholesky decomposition

Why Cholesky, Why Direct Form?

- G is Hermitian positive definite $\rightarrow G = LL^H$ exists, Cholesky exhibits better numerical stability and lower complexity than QR/LDL.
- Compared to equation solving (solve per symbol), direct form reuses W_k in block-fading channel assumption.
- $C_{\text{direct}} < C_{\text{eq-solv}}$ for $N < 14$, which fits practical MIMO

Amortizing with Block-Fading channel Assumption

- W_k is compute once per block, and applied to all the following L CP-OFDM symbols in the block
- $C_{\text{direct}} = \underbrace{\left(\frac{5}{3}N^3 + 2N^2 - \frac{2}{3}N\right)K}_{\text{once per block}} + \underbrace{N^2 KL}_{\text{once per block}}$

Implementation: Complex Arithmetic and Register Management

Complex arithmetic on integer RVV:

- **I/Q split:** real and imaginary parts in separate registers
- **Format:** 32-bit data + 64-bit accumulators
Explicit managed by vnclip at each stage output
- **Complex MAC:** $s = a + bc$
 $s_r = \text{vsub}(\text{vwmac}(a_r, b_r, c_r), \text{vwmul}(b_r, c_i))$
 $s_i = \text{vwmac}(\text{vwmac}(a_r, b_r, c_i), b_i, c_r)$
 $\rightarrow 4$ to 6 integer ops per complex MAC
- **Hermitian free:** $H_{ij}^H = H_{ji}^*$, through index transform which has same op count

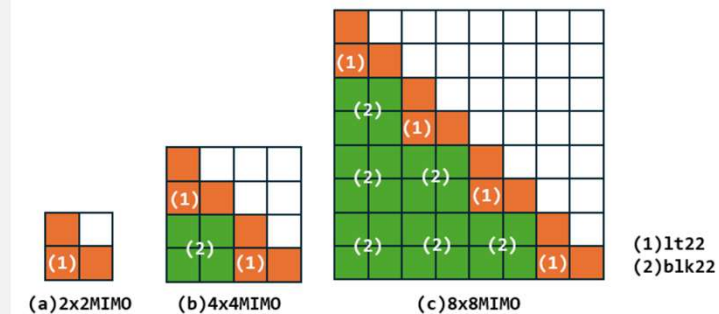
RVV 1.0 functionally supports complex fixed-point arithmetic, while the operations have to be explicitly managed.

Vector Register Management:

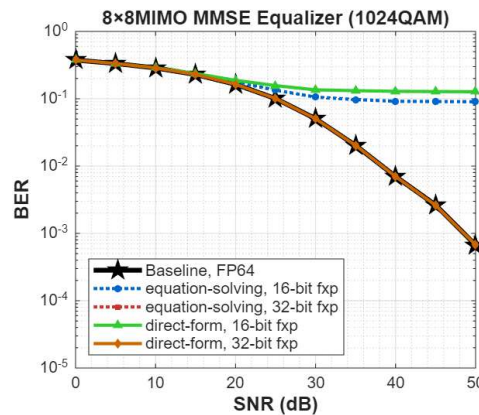
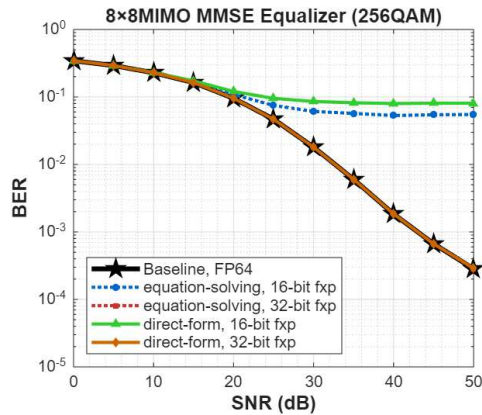
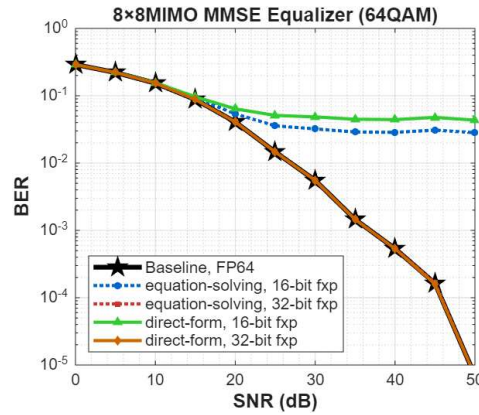
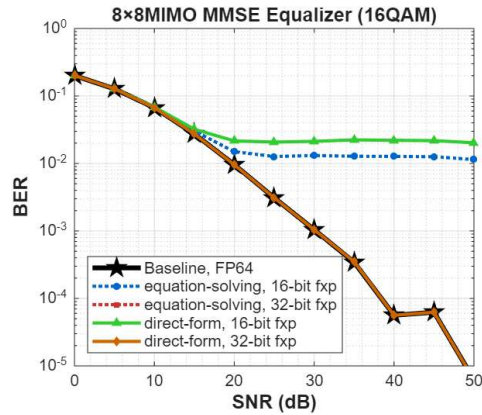
- I/Q separated complex format results in registers pressure
- Spill and refill of vector register occupies memory bandwidth and further degrades kernel throughput

Budget: 32 architectural vector registers (LMUL=1)

Blocking: Gram partitioned into 2×2 tiles — lt22 (diagonal) / blk22 (off-diagonal), live register set stays < 32, while data locality improves, ensuring no spills or refills.



Fixed-point Numerical Analysis of the MMSE Equalizer



32-bit format for fixed-point MMSE Equalizer on RVV:

- Both the equation solving and the direct-form MMSE equalizer, the error-rate performance of 16-bit implementation does not improve with increasing input SNR when input SNR > 25dB.
- Noise floor results from lacking of accumulator headroom bits and quantization error with 16-bit format.
- The 32-bit fixed-point implementation achieves error-rate performance equivalent to the FP64 baseline for 16-, 64-, 256-, and 1024-QAM.

Cycle-Performance Across Different RVV Platforms

MIMO	Algorithm Step	Ara Platform			Saturn Platform			Speedup (Saturn vs. Ara2)
		CVA6 (scalar host)	Ara2 (RVV)	Speedup (RVV / scalar)	Shuttle (scalar host)	Saturn (RVV)	Speedup (RVV / scalar)	
4 × 4	Compute Gram matrix	1091	10.61	53×	635	5.52	101×	2.5×
	Cholesky decomposition	467	7.29		417	2.72		
	Forward substitution	762	22.28		587	6.96		
	Backward substitution	1068	22.36		802	8.79		
	Spatial Filtering	372	7.52	49×	339	4.07	83×	1.85×

1. The RVV platforms are configured with 4096-bit VLEN and 512-bit/cycle DLEN.
2. The RISC-V scalar host cores are configured as RV64I (64-bit/cycle).
3. The unit is defined as the average of cycles consumed to compute a received in a resource element

- The kernel throughput depends strongly on the microarchitectural factors
 - Ara2 provides a **256-bit/cycle** vector load and store unit (VLSU).
 - Saturn keeps the VLSU throughput same as data path, which is **512-bit/cycle**.
- Algorithm parts with nested data dependencies benefits from sufficient memory band width.

Real-time Requirements on the Processor Frequency (MHz)

	LTE-High End 20MHz@15kHz			5G FR1-Mid n78(3.5GHz)@100MHz@30kHz		
	weight matrix	spatial filtering	total	weight matrix	spatial filtering	total
	2					
ARA	51.68	136.63	188.32	274.96	726.88	1001.84
Saturn	20.82	94.57	115.39	110.77	503.09	613.86
	4					
ARA	300.10	505.44	805.55	1596.55	2688.96	4285.51
Saturn	115.13	273.53	388.65	612.47	1455.15	2067.62
	8					
ARA	1893.80	1766.10	3659.90	10075.00	9395.65	19470.65
Saturn	775.10	804.96	1580.05	4123.52	4282.37	8405.88

- $f_{required}(MHz) = NR \times N_{cycles} \times L_{slot} \times N_{RB} \times N_{subcarrier} / T_{slot}(TTI) \times \eta$
 - Real-time requirements on processor frequency to process the resource elements within a slot.
 - Redundancy factor η is set to 15%.
- Referenced synthesis results:
 - ARA2: GlobalFoundries 22FDX, **1350MHz**
 - Saturn: TSMC 16nm, **800MHz**

Conclusion and Further Research

Conclusions:

- **Standard RVV is functionally sufficient to support real-time 2×2 and 4×4 MIMO equalization** in LTE and 5G FR1 scenarios.
- **BER-drive format selection:** 16-bit fixed point floors above ~25 dB SNR, while 32-bit Q10.20 matches the FP64 baseline across 16 to 1024 QAM
- **Microarchitecture decisive for kernel throughput**, which exhibits a 2.5× difference between two RVV-compliant cores.
 - Same kernel and hardware configuration on VLEN, DLEN
 - Different VLSU throughput, chaining quality, short-vector dependency tolerance.

Further Research Aspects:

- The standard RVV ISA **lacks support for DSP-specific primitives and microarchitectures**
 - Primitives for fixed-point complex arithmetic to reduce the register pressure and code size
 - Fast in-register table lookup, which support gather at beat rate by window-sized crossbar network
- Learn from use cases: **Extend RVV ISA for wireless baseband processing kernels**
 - Optimizations based on standard RVV-based kernels and uses them as benchmarks