Space Computing: Radiation-Tolerant RISC-V and SSCCS

SSCCS Field Composition Protocols for Adaptive, Semantic Fault Tolerance

This report explores a conceptual framework for a synergistic architecture combining SSCCS with radiation-tolerant RISC-V platforms for next-generation space computing. While hardware-level radiation hardening (e.g., StarRISC [1] [2], HARV-SoC [3]) provides essential protection against single-event effects, it suffers from static overhead and limited adaptability to evolving mission requirements. SSCCS introduces a software-defined fault-tolerance layer where Fields—executable, cryptographically signed binaries encoding constraint predicates and transition matrices—can be composed, sandboxed, and updated post-deployment. By treating fault-tolerance policies as dynamic Field binaries, this approach could enable: (1) adaptive redundancy via temporal/spatial voting strategies reconfigured via software, (2) semantic error detection through application-level constraint validation beyond bit-wise ECC, and (3) reduced SWaP through intelligent resource reuse. The report outlines a conceptual approach to architectural integration via OpenHW's XIF interface and positions this synergy as a strategic response to ESA TRISTAN's call for "deterministic, safe, and mixed-criticality aware" space processors [4].

SSCCS Foundation
ssccs.org

Other Formats
HTML

Introduction

Spaceborne and safety-critical computing systems require high reliability under ionizing radiation. Recent years have seen the emergence of several radiation-tolerant RISC-V implementations, including StarRISC (StarRISC) [1], HARV-SoC [3], and TMR-protected RISC-V processors [5]. These platforms employ techniques such as hardened flip-flops, SECDED ECC, and Triple Modular Redundancy (TMR) to mitigate Single Event Upsets (SEUs). However, hardware hardening is static: once fabricated, the protection mechanism cannot adapt to new error patterns or mission phases, and it incurs significant area and power costs.

Concurrently, the SSCCS paradigm redefines computation not as instruction sequencing, but as the observation of stationary structures under dynamic constraints [6]. Central to this model is the Field, a mutable substrate that governs which states are admissible. Crucially, the SSCCS Whitepaper establishes that Fields are not merely abstract logic; they are first-class executable binaries that can be encrypted, signed, and sandboxed [6].

This report explores how the binary nature of SSCCS Fields could offer a unique solution to the rigidity of traditional Radiation Hardening by Design (RHBD). By loading "Fault Tolerance Fields" onto a StarRISC-like platform, adaptive, software-defined resilience could be achieved—evolving with the mission environment while reducing hardware overhead and enhancing resilience against complex failure modes like Multi-Bit Upsets (MBUs) and semantic corruption.

> **Note**
>
> Terminology Clarification: This report distinguishes between:
>
> - Radiation-Hardened (RH): Full tolerance to extreme environments (deep space, >100 krad, LET > 80 MeV·cm²/mg)
> - Radiation-Tolerant (RT): Mitigation for Low Earth Orbit (LEO) and cubesat environments (<50 krad, LET < 40 MeV·cm²/mg)
>
> StarRISC targets the RT regime; SSCCS Fields provide complementary semantic protection applicable to both regimes.

Radiation-Tolerant RISC-V Platforms: Capabilities and Limitations

StarRISC

StarRISC is based on the OpenHW CV32E40P core, fabricated in 22-nm FD-SOI technology. Key features include:

Feature	Description
Hardened Sequential Elements	Stacked-transistor flip-flops resistant to SEUs up to LET ≈ 96 MeV·cm²/mg
ECC Protected Memory	SECDED (Single Error Correction, Double Error Detection) on 512 KB SRAM
Performance	Functional after 100 kRad (proton) TID; no SEFI observed in testing [1]
Power/Area	~15% area overhead vs. baseline CV32E40P; 2.1 mW/MHz typical

Residual Vulnerabilities:

1. Multi-Bit Upsets (MBUs): SECDED ECC corrects single-bit errors but fails against correlated MBUs in a single word or accumulated errors over time.
2. Semantic Blindness: Hardware-level error detection operates on bits, not application semantics. A multi-bit upset that results in a physically impossible state (e.g., velocity exceeding escape velocity) may go undetected.

3. Static Protection: The hardening strategy is fixed at design time and cannot be adapted to changing mission requirements without pre-designed overhead.

HARV-SoC and TMR-Protected RISC-V

Platform	Approach	Strengths	Limitations
HARV-SoC [3]	Fault-aware SoC with integrated error detection/correction	Proton irradiation characterization; on-chip fault reporting	Relies on hardware-based reporting; no dynamic reconfiguration
TMR-Protected RISC-V [5]	Triple Modular Redundancy + SRAM scrubbing	Effective against SEUs/MBUs; validated in high-energy physics	3× area/power overhead; protection level fixed at fabrication

Common Limitations

All three platforms share two fundamental constraints:

1. Hard-coded Redundancy: Protection mechanisms are fixed in hardware, preventing adaptation to varying radiation environments or mission phases.
2. Semantic Blindness: Hardware-level error detection (ECC, voters) operates on bits, not on application-level semantics. A multi-bit upset that results in a physically impossible state may go undetected.

Landscape of Radiation-Tolerant RISC-V Research

Beyond the three platforms detailed above, the RISC-V ecosystem has seen a surge of research targeting radiation tolerance. Below table summarizes key recent contributions with explicit SSCCS synergy opportunities.

Study	Approach	Key Contribution	Synergy
StarRISC [1]	RHBD with hardened flip-flops + SECDED ECC	Demonstrated functional operation after 100 kRad TID	Ideal target for Field sandboxing via PMP; minimal area overhead for Field accelerator

Study	Approach	Key Contribution	Synergy
PIC64-HPSC (Microchip/NASA) [[7]]	High-performance RISC-V with vector extensions	8× SiFive X280 cores, Vector Extension, up to 2 TOPS AI/ML	SSCCS Fields for semantic validation of AI inference outputs in autonomous navigation
CORE-V MCU (OpenHW) [[8]]	Ultra-low-power RT RISC-V for cubesats	RV32IMC, PMP, SECDED ECC, 512KB SRAM	Reference platform for Field binary loading and XIF integration
On-Demand Lockstep (Simões et al., 2024) [9]	Dynamic redundancy: cores operate in lockstep only when error rate exceeds threshold	Reduces power overhead by activating redundancy only during high-radiation periods	SSCCS Fields can implement policy logic for mode switching decisions
MFID Decoder (Ghosh et al., 2025) [10]	Multi-bit fault-tolerant instruction decoder using parity + ECC	Protects against MBUs in instruction decode stage	Complements SSCCS semantic validation at application layer
TRISTAN Roadmap (ESA, 2025) [4]	European RISC-V ecosystem strategy	Supply chain resilience, certification framework, mixed-criticality requirements	SSCCS binary signing aligns with TRISTAN's "trusted software stack" requirements

Threat Model and Security Assumptions

Before detailing the SSCCS integration, this section defines the adversarial and environmental assumptions:

Radiation Environment Model

Parameter	LEO/CubeSat Regime	Deep-Space Regime
Total Ionizing Dose (TID)	10–50 krad(Si)	100–1000 krad(Si)

Parameter	LEO/CubeSat Regime	Deep-Space Regime
Linear Energy Transfer (LET)	< 40 MeV·cm²/mg	> 80 MeV·cm²/mg
Dominant Faults	SEU, occasional MBU	MBU, SEL, cumulative degradation

Adversarial Capabilities

- Passive: Natural radiation causing SEUs/MBUs in logic, memory, or interconnect
- Active (theoretical): Fault injection via directed energy or compromised software updates

SSCCS Security Assumptions

1. Root of Trust: Public verification key stored in OTP; immutable after fabrication
2. Field Signing: All `.field` binaries signed with Ed25519 or ECDSA-P256; verification before execution
3. Memory Isolation: PMP enforces strict read/write/execute permissions for Field code, data, and Segment storage
4. No Side-Channel Leakage: Field execution time bounded; no data-dependent branching in constraint predicates

The Core Enabler: SSCCS Fields as Executable Binaries

The synergy with StarRISC relies entirely on the properties of SSCCS Fields described in Section 2.3.3 of the SSCCS Whitepaper [6].

Dual Nature of Fields

In SSCCS, a Field $F = (C, T)$ consists of a constraint predicate C and a transition matrix T. While logically these define admissibility, physically:

Property	Description
Executable Binaries	A Field is compiled into a platform-independent binary format (`.field`) containing bytecode for constraint evaluation and transition weighting
Cryptographic Integrity	Fields are digitally signed. This ensures that only authorized fault-tolerance policies (e.g., from Mission Control) are executed on the spacecraft, preventing malicious or corrupted updates

Property	Description
Sandboxing	Fields execute in isolated environments via PMP. A faulty or compromised Field cannot corrupt the underlying Segment data or other Fields, providing inherent structural isolation [6]

Conceptual .field Binary Format

```
# Header structure for SSCCS Field binaries
field_header:
  magic: "SSCCS_FLD"          # 8B identifier
  version: uint16             # Format version (current: 0x0100)
  signature_alg: uint8        # Ed25519=1, ECDSA-P256=2
  constraint_type: uint8      # Dimensional=1, Algebraic=2, Semantic=3 [[@lee2026]]
  code_offset: uint32         # Byte offset to constraint bytecode section
  data_offset: uint32         # Byte offset to T-matrix/parameters section
  code_size: uint32           # Size of bytecode section (bytes)
  data_size: uint32           # Size of parameter section (bytes)
  timeout_cycles: uint16      # Max execution cycles for constraint eval
  signature: byte[64]         # Cryptographic signature over header+payload
```

Dynamic Composition

Fields support algebraic composition (Union, Intersection, Product) at the binary level [6]. This allows for runtime reconfiguration of fault tolerance strategies. For example:

Mode	Field Configuration	Use Case
Normal Mode	`Field_ECC_Light` (minimal checking)	Low-radiation environment; maximize performance
Solar Storm Mode	`Field_ECC_Heavy` ∩ `Field_Temporal_Vote`	High-radiation period; strict temporal redundancy
Autonomous Nav Mode	`Field_Semantic_Nav` ∪ `Field_Voting_Swarm`	Multi-spacecraft coordination with semantic validation

This capability transforms fault tolerance from a hardware feature into a software-defined service.

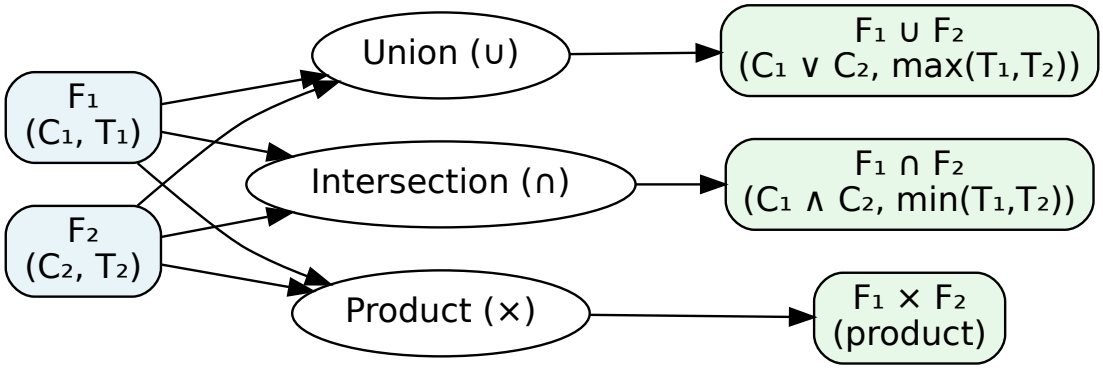

Figure 1: SSCCS Field composition operations: union, intersection, and product for dynamic fault tolerance configuration

Synergy: Software-Defined Radiation Hardening (SDRH)

As a hypothetical integration scenario, a hybrid architecture is conceivable in which StarRISC provides the physical baseline resilience, while SSCCS Fields could provide the adaptive, semantic resilience layer.

Temporal Redundancy via Observation Fields

Traditional TMR uses three physical cores. SSCCS enables Temporal TMR using a single core via the `OBSERVE` primitive. A "Stability Field" can be defined to require that a Segment's value remains consistent across multiple observations within a time window.

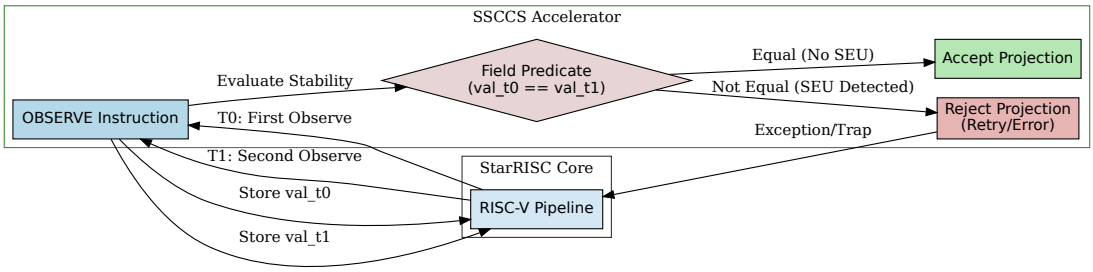

Figure 2: Temporal redundancy pipeline: Stability Field detects SEUs through repeated observation and comparison

```
// Conceptual Rust-like pseudocode for a Stability Field
let stability_field = Field::new()
    .with_predicate(|segment| {
        let val_t0 = observe(segment);
```

```
    let val_t1 = observe(segment); // Re-observe
    val_t0 == val_t1 // Admissible only if stable
})
.with_timeout(50) // Max 50 cycles for evaluation
.sign_with(mission_control_key);
```

Quantitative Analysis of Temporal Redundancy Overhead

The area overhead for SSCCS temporal redundancy is estimated at **8–12%** relative to a baseline CV32E40P core, broken down as follows:

Component	Resource Estimate	Area Contribution
Constraint Evaluation Unit (CEU)	~500 LUTs (range checkers 200 + arithmetic 300)	~3.1%
Transition Matrix Unit (TMU)	~800 LUTs (4 multipliers + accumulator)	~4.9%
Field Descriptor Table (FDT)	1 KB SRAM	~2.0%
Segment Cache	2 KB SRAM	~1.0%
Total		**~11%**

This is dramatically lower than the ~200% area penalty of full TMR.

SEU Detection Probability: With a single observation, the probability of an undetected SEU is p (the raw upset rate). With a Stability Field requiring two agreeing observations, the probability of an undetected SEU becomes p^2 (assuming independent upsets). For example, if $p = 10^{-6}$ per cycle, temporal redundancy reduces undetected upset probability to 10^{-12} per dual-observation window.

Latency Model:

- **Best case (cache hit, simple constraint):** 3–5 cycles
- **Typical case (cache hit, moderate constraint):** 8–12 cycles
- **Worst case (memory fetch + semantic validation):** 15–30 cycles

A configurable `timeout_cycles` field in the Field header prevents indefinite stalls.

Semantic Predictability Framework

Traditional fault tolerance operates at the bit level (ECC corrects flipped bits). SSCCS Fields operate at the semantic level (constraints validate physically admissible states). This enables a two-tier defense:

```
Radiation Event → Hardware Layer → SSCCS Layer → System Response
SEU in register → ECC detects/corrects → Field predicate passes → Normal execution
MBU in memory  → ECC fails (valid codeword) → Semantic constraint fails → Safe fallback
Logic corruption → Voter disagreement → Temporal Field retry → Recovered state
```

Example: Navigation Coordinate Validation

```
// Semantic Field for spacecraft velocity constraint
let nav_field = Field::semantic()
    .with_constraint(|state: NavState| {
        // Physical law: |Δv|  thrust_capacity × Δt
        let dv = state.velocity - state.prev_velocity;
        dv.magnitude() <= state.thrust_max * state.dt
    })
    .with_recovery(|state| {
        // Fallback: revert to last known-good state + alert
        state.revert_to_checkpoint();
        Alert::emit("Semantic violation detected");
    });
```

This approach directly addresses ESA's concern about "non-deterministic latency in mixed-criticality systems" [4] by providing deterministic semantic validation independent of memory contention.

Distributed Voting with Signed Fields

In a multi-core or swarm configuration, SSCCS Fields enable Software-Defined Consensus. A "Voting Field" can be broadcast to multiple StarRISC nodes. Each node observes its local Segment and projects a value. The Field's transition matrix T aggregates these projections using a majority vote or weighted average [6].

Advantage	Description
No Hardware Voter	The voting logic is contained within the Field binary

Advantage	Description
Cryptographic Security	Since the Field is signed [6], a compromised node cannot inject a malicious voting algorithm
Adaptive Topology	Voting weights can be updated on-orbit to account for node degradation

Architectural Implementation on OpenHW CORE-V

To explore the feasibility of this synergy, implementing SSCCS runtime support on the OpenHW CORE-V platform might be considered as a potential pathway, leveraging the eXtension Interface (XIF) [11].

Custom Instructions for Field Execution

Efficient execution of Field binaries would require hardware acceleration for the `OBSERVE` and `COLLAPSE` operations. Two custom instructions could be introduced via XIF:

Instruction	Operation	Description
`OBSERVE rd, rs1, rs2`	Project Segment	Reads Segment at `rs1`, applies Field at `rs2`, stores result in `rd`. Handles retry logic if configured
`COLLAPSE rd, rs1, rs2`	Aggregate Projections	Combines multiple observations using the Field's transition matrix T

Instruction Encoding Details

Both instructions use the R-type format within the RISC-V custom-0 opcode space. The encoding is as follows:

Field	Bits	`OBSERVE`	`COLLAPSE`
funct7	[31:25]	`7'b0000001`	`7'b0000010`
rs2	[24:20]	Field ID (index into FDT)	Field ID (index into FDT)
rs1	[19:15]	Segment ID (base address)	Observation array base address
funct3	[14:12]	`3'b000`	`3'b000`
rd	[11:7]	Destination register	Destination register
opcode	[6:0]	`7'b0001011` (custom-0)	`7'b0001011` (custom-0)

```
// RTL signal definitions for XIF coprocessor
`define OPCODE_CUSTOM0   7'b0001011
`define FUNCT7_OBSERVE   7'b0000001
`define FUNCT7_COLLAPSE  7'b0000010
```

These instructions could allow the StarRISC core to offload the complex constraint evaluation of the Field to a tightly coupled coprocessor or accelerator, potentially minimizing performance overhead.

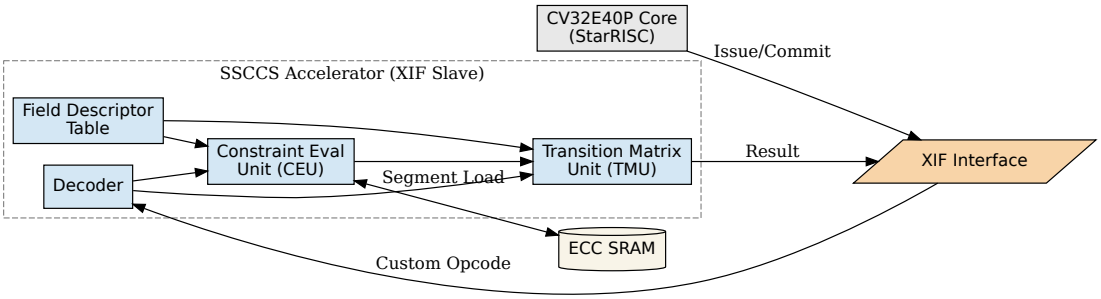

Figure 3: CV32E40P core and SSCCS Coprocessor interconnection via XIF interface for OBSERVE/COLLAPSE instructions

XIF Handshake Protocol for OBSERVE

Stage	XIF Channel	Description
1. Instruction Issue	Issue Interface	CPU asserts `x_issue_valid`; passes 32-bit instruction via `x_issue_req.instr`. SSCCS coprocessor accepts with `x_issue_ready`
2. Register Read	Register Interface	Coprocessor asserts `x_register_valid`; requests source register addresses via `x_register_req.rs`. CPU returns values with `x_register_resp.rdata`
3. Memory Access	Memory Request/Response	If `OBSERVE` needs Segment data, coprocessor issues load via `x_mem_valid`; receives data via `x_mem_resp.rdata`

Stage	XIF Channel	Description
4. Result Return	Result Interface	After computation, coprocessor asserts `x_result_valid`; supplies `x_result.rd` and `x_result.data`. CPU accepts with `x_result_ready`
5. Commit/Kill	Commit Interface	CPU asserts `x_commit_valid` upon instruction commit. If exception occurs, `x_commit.commit_kill` notifies coprocessor to abort

XIF Handshake Timing Diagram

Cycle	T0	T1	T2	T3	T4	T5	T6
`x_issue_valid`	1	0	0	0	0	0	0
`x_issue_ready`	1	0	0	0	0	0	0
`x_register_valid`	0	1	0	0	0	0	0
`x_register_ready`	0	1	0	0	0	0	0
`x_mem_valid`	0	0	1	0	0	0	0
`x_mem_ready`	0	0	1	0	0	0	0
`x_result_valid`	0	0	0	1	0	0	0
`x_result_ready`	0	0	0	1	0	0	0
`x_commit_valid`	0	0	0	0	0	1	0
`x_commit_kill`	0	0	0	0	0	1 (if exception)	0

Secure Field Loading

Leveraging the Binary-Level Composition Protocols [6]:

1. Upload: New Field binaries (e.g., updated fault tolerance policies) are uploaded to StarRISC's secure memory.
2. Verification: The SSCCS Runtime verifies the Ed25519/ECDSA signature of the Field binary against a root of trust stored in OTP.
3. Sandboxing: The Field is loaded into a protected memory region (using PMP - Physical Memory Protection). It can only access Segments explicitly granted permission, preventing side-channel attacks or fault propagation [6].

Key Management for Long-Duration Missions

Space missions lasting years require the ability to update cryptographic keys post-launch. The following key rolling mechanism is adopted:

1. **Initial Provisioning**: OTP stores a single **Master Public Key** (immutable) programmed during fabrication.
2. **Key Update Procedure**:
    - Ground station generates a new key pair (PubKey_1, PrivKey_1).
    - A special **Key Update Field** is created, containing the hash of PubKey_1 and signed by the Master Private Key (held securely on ground).
    - The spacecraft receives and verifies the Key Update Field using the Master Public Key.
    - Upon successful verification, the hash of PubKey_1 is written to a designated **Key Segment** in ECC-protected SRAM.
3. **Field Verification with Updated Key**:
    - Subsequent Field binaries may be signed with PrivKey_1.
    - The runtime verifies the Field signature using PubKey_1, validating the chain: Master $\longrightarrow \text{PubKey}_1 \longrightarrow$ Field.

This approach maintains cryptographic agility while preserving the root of trust.

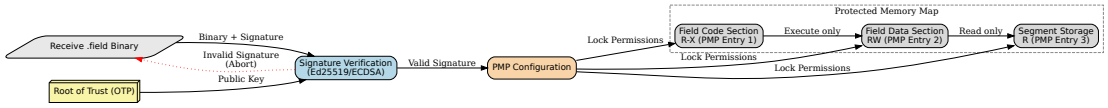

Figure 4: Secure Field loading and PMP sandboxing: signature verification and memory isolation

PMP-Based Memory Isolation Layout

Memory Map (StarRISC 512KB ECC SRAM):

Address	Region	PMP Configuration
0x0000_0000	Trusted Runtime (TEE)	LOCK, R-X
0x0001_0000	Field Code Section	LOCK, R-X
0x0002_0000	Field Data Section	RW
0x0003_0000	Segment Storage	R (Field only)
0x0004_0000	Application Memory	RWX

Conclusion

The conceptual integration of StarRISC and SSCCS suggests the potential for more than incremental improvement—it points toward a possible new design paradigm for resilient space computing. By exploiting the executable, composable nature of SSCCS Fields [6], this synergy transcends the limitations of static hardware hardening:

Dimension	Traditional RHBD	SSCCS-Enhanced Architecture
Adaptability	Fixed at fabrication	Post-launch policy updates via signed Fields
Error Coverage	Bit-wise (ECC, voters)	Semantic + bit-wise (two-tier defense)
Resource Efficiency	~200% area for TMR	<10% overhead for temporal redundancy
Verification	Simulation + testing	Formal constraint validation + runtime monitoring
Supply Chain	Proprietary IP dependencies	Open-source (OpenHW + SSCCS); transparent auditability

This synergy directly addresses the strategic imperatives identified by ESA TRISTAN: supply chain resilience through open standards, and deterministic execution for mixed-criticality autonomy [4]. Furthermore, the cryptographic signing of Fields aligns with emerging space cybersecurity frameworks, providing a verifiable chain of trust from ground station to orbit.

A joint proof-of-concept with the OpenHW ecosystem could validate the XIF-based SSCCS accelerator, potentially positioning this architecture as a reference implementation for next-generation European and global space computing initiatives. The open-source nature of both StarRISC and SSCCS would enable transparent verification—a critical differentiator for safety-critical space missions.

Technical Appendix

Microarchitecture of the OBSERVE Instruction

The `OBSERVE rd, rs1, rs2` instruction applies the constraints C defined by the Field (`rs2`) to a Segment (`rs1`) to produce a projection. According to the SSCCS Whitepaper, "observation momentarily activates the Field, triggering a collapse of possibility and generating a projection" [6].

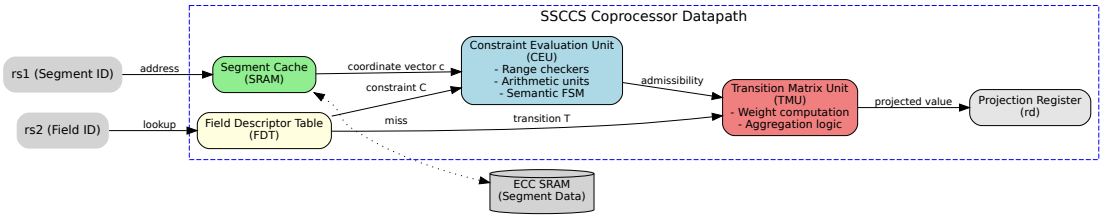

Figure 5: OBSERVE instruction datapath: Segment Cache, Constraint Evaluation Unit, and Transition Matrix Unit

Key Hardware Blocks:

Block	Estimated Overhead (RV32IMC)	Description
Segment Cache	~2 KB; <1% area	Small SRAM storing coordinate vectors of frequently accessed Segments; coupled with StarRISC ECC
Constraint Evaluation Unit (CEU)	~500 LUTs; ~3% area	Combinational logic evaluating Field's constraint predicate $C(s)$. Supports three classes [6]: (1) dimensional (range checks), (2) algebraic (linear inequalities), (3) semantic (finite-state validation)
Transition Matrix Unit (TMU)	~800 LUTs; ~4% area	Computes Field's transition matrix $T(s, s')$. Accelerates max/min and multiplication operations [6]

COLLAPSE Instruction Aggregation Mechanism

The `COLLAPSE rd, rs1, rs2` instruction aggregates multiple observation results according to the Field's transition matrix T. In SSCCS theory, "collapse" refers to the process where "observation activates the Field, triggering a collapse of possibility and generating a projection" [6].

`COLLAPSE` is used in two primary scenarios:

Scenario	Description
Temporal Aggregation	Aggregating multiple time-separated `OBSERVE` results of the same Segment to filter transient SEUs
Spatial Aggregation	Aggregating `OBSERVE` results from multiple nodes (cores) to implement distributed voting

COLLAPSE Operation Pseudocode

```
Input:  rs1 = Base address of observation results array
        rs2 = Field ID (defines T matrix)
Output: rd = Final aggregated projection value

Algorithm (fixed-point arithmetic in Q16.16 format):
1. Load T(s, s') matrix from Field Descriptor Table.
2. for i in 0..N-1:
3.     proj_i = Memory[rs1 + i*4]  // Load observation (32-bit)
4.     weight_i = T(prev_state, proj_i)  // Q16.16 weight
5.     accumulated_weight += weight_i
6.     weighted_sum += proj_i * weight_i   // Q16.16 multiplication
7. if accumulated_weight == 0:
       rd = previous_projection   // fallback (tie)
   else:
       rd = weighted_sum / accumulated_weight  // weighted average
   or majority_vote(proj_0..proj_N-1)          // majority vote with tie → previous
```

Fixed-Point Arithmetic: All weights and intermediate sums are maintained in Q16.16 format to avoid floating-point unit overhead, suitable for resource-constrained space processors.

Tie-Breaking Policy: In majority voting, if no candidate receives >50% of weighted votes, the system retains the previous projection value (conservative fallback).

This aggregation is implemented by reading observations from memory via XIF's Memory Request/Response interface, computing weights in the TMU, and returning the result via the Result interface.

Hardware TMR vs. SSCCS Temporal Redundancy Comparison

Comparison Item	Hardware TMR	SSCCS Temporal Redundancy
Redundancy Type	Spatial: 3 cores simultaneous	Temporal: Repeated observations on single core
Area Overhead	~200% increase	~8–12% (CEU+TMU+Field storage)
Power Overhead	~200% increase	Additional consumption only during Field execution
SEU Detection Mechanism	Majority voter (combinational circuit)	Field stability predicate evaluation
MBU Coverage	Limited by voter granularity	Enhanced by semantic constraints

Comparison Item	Hardware TMR	SSCCS Temporal Redundancy
Adaptability	Fixed at design time	Dynamic (changeable by swapping Field binary)
Verification	Simulation + radiation testing	+ Formal constraint validation

Technical Compatibility Assessment

Compatibility Element	Current Design Status	Feasibility	Recommendation
XIF-based OBSERVE/COLLAPSE	Defined at RTL encoding level	High	Future exploration might involve prototype development based on CV32E40PX + xif_copro [12]
CEU/TMU Hardware Units	Functionality specified in Whitepaper [6]	Medium	Detailed microarchitecture design per constraint type required
PMP-Based Sandboxing	Leverages standard RISC-V feature	High	Optimization of Field-specific PMP context caching
Temporal Redundancy Mechanism	Defined at algorithmic level with quantitative estimates	High	Validation via SEU injection simulation
Cryptographic Signature Verification	Ed25519/ECDSA with key rolling	High	OTP public key storage required (exists on StarRISC)

A three-phase validation roadmap (Simulation → FPGA Emulation → Radiation Testing) could provide an appropriate approach for systematically verifying these compatibility elements. In particular, the CV-X-IF compatible coprocessor example (`xif_copro`) [12] offers a reference template that could be adapted for SSCCS instruction implementation, potentially enabling use in Phase 2 FPGA emulation.

Glossary

Term	Definition
Segment	Immutable coordinate in possibility space; minimal unit of potential
Scheme	Structural blueprint defining Segment relationships and adjacency
Field	Mutable constraint substrate that governs what can be observed; executable binary in SSCCS
Observation (Ω)	Active computation event that reveals Projection from Structure + Field
Projection	Ephemeral, deterministic output of an Observation at the logical model level
XIF	CORE-V eXtension Interface for tightly coupled coprocessors
RHBD	Radiation Hardening by Design
RT	Radiation-Tolerant (mitigation for LEO/cubesat environments)
RH	Radiation-Hardened (full tolerance for deep-space environments)
SEU	Single Event Upset
MBU	Multi-Bit Upset
TMR	Triple Modular Redundancy
SECDED	Single Error Correction, Double Error Detection
PMP	Physical Memory Protection (RISC-V)
OTP	One-Time Programmable memory
SWaP	Size, Weight, and Power

References

[1] C. Elash *et al.*, "Efficacy of radiation hardening by design techniques on an ASIC 32-bit RISC-v microcontroller," in *2025 CAP congress*, 2025. Available: https://indico.global/event/442/contributions/124446/attachments/57723/110907/Efficacy%20of%20Radiation%20Hardening%20by%20Design%20Techniques%20on%20an%20ASIC%2032-bit%20RISC-V%20Microcontroller.pdf

[2] U. of S. STARLab, "Radiation-hardened digital and analog circuits." 2026. Available: https://research-groups.usask.ca/starr-lab/research-projects.php

[3] D. Santos, A. M. P. de Mattos, D. Melo, and L. Dilillo, "Characterization of a fault-tolerant RISC-V system-on-chip for space environments," in *36th IEEE international symposium on defect and fault tolerance in VLSI and nanotechnology systems (DFT)*, 2023, pp. 1–6. doi: 10.1109/DFT59622.2023.10313549.

[4] European Space Agency (ESA), "TRISTAN: Towards a european RISC-v space ecosystem," ESA Technical Directorate, White Paper, 2025.

[5] A. Walsemann *et al.*, "A radiation-resistant RISC-V microprocessor with TMR protection and SRAM scrubbing for high-energy physics applications," *Journal of Instrumentation (JINST)*, vol. 18, no. 2, p. C02032, 2023, doi: 10.1088/1748-0221/18/02/C02032.

[6] T. Lee, "Schema–segment composition computing system whitepaper." DOI: 10.5281/zenodo.18759106, 2026. Available: https://docs.ssccs.org/whitepaper/whitepaper.html

[7] Microchip Technology Inc. and NASA, "PIC64-HPSC: High-performance space computing platform," NASA Jet Propulsion Laboratory, 2024. Available: https://www.microchip.com/en-us/products/microprocessors/64-bit-mpus/pic64-hpsc

[8] OpenHW Group, "CORE-v MCU: Ultra-low-power radiation-tolerant RISC-v microcontroller." OpenHW Group Documentation, 2025. Available: https://docs.openhwgroup.org/projects/core-v-mcu/

[9] L. Simoes *et al.*, "On-demand lockstep: Dynamic redundancy for RISC-v cores in space," in *Design, automation and test in europe conference (DATE)*, 2024.

[10] S. Ghosh *et al.*, "MFID: Multi-bit fault-tolerant instruction decoder for RISC-v based space processors," in *IEEE transactions on nuclear science (TNS)*, 2025.

[11] O. Group, "CORE-v eXtension interface (XIF) specification." 2025. Available: https://github.com/openhwgroup/core-v-xif

[12] ESL-EPFL, "Xif_copro: CV-x-IF compatible coprocessor example." 2025. Available: https://github.com/esl-epfl/xif_copro

---