Resonant Computing: Coherence Processing Units and Non-Boolean Logic

Author: Hadi Servat

Affiliation: Independent Researcher

Contact: publish@fractalresonance.com

Website: fractalresonance.com

DOI: 10.5281/zenodo.18091126
Permanent record: https://doi.org/10.5281/zenodo.18091126 · PDF: https://zenodo.org/records/18091126

Abstract

Proposes a computing architecture based on coupled oscillators rather than logic gates. Introduces the Resonant Bit (R-bit), which stores information in phase relations to a global coherence field, and the Coherence Processing Unit (CPU), which computes via deterministic phase-locking dynamics. We analyze error correction, computational complexity, and advantages over both classical and quantum computing paradigms.

1. Introduction

Digital computing is built on Boolean logic: bits are 0 or 1, gates compute AND, OR, NOT. Quantum computing extends this with superposition: qubits are $\alpha|0\rangle + \beta|1\rangle$, and gates are unitary matrices. Both paradigms treat computation as discrete state manipulation.

This paper proposes a third paradigm: **Resonant Computing**. Instead of manipulating discrete states, a resonant computer manipulates phase relationships between coupled oscillators. The fundamental unit is not a bit or qubit but an **R-bit** (Resonant bit)—an oscillator whose phase relative to a global clock encodes information. Computation occurs not through gate operations but through **phase-locking dynamics** governed by the Lambda-field.

The key insight: if quantum mechanics is fundamentally about coherence dynamics ([[FRC-100-001]]), then a computer that directly exploits coherence dynamics should be more natural—and potentially more powerful—than one that fights decoherence (quantum computing) or ignores it entirely (classical computing).

Resonant Computing Concept

Resonant Computing Concept

2. The Resonant Bit (R-bit)

2.1 Definition

An R-bit is an oscillator with well-defined frequency ω_0 and controllable phase ϕ:

\[\text{R-bit state:} \quad \phi \in [0, 2\pi)\]

Unlike a classical bit (2 states) or qubit (continuous but fragile), an R-bit stores a continuous phase that is:

- **Robust:** Phase-locking to the global clock provides natural error correction
- **Continuous:** The full $[0, 2\pi)$ range is accessible
- **Relational:** Information is encoded in phase *differences*, not absolute values

2.2 Information Encoding

Information is stored in the phase relation between R-bits and the global Lambda-field clock:

\[\text{Information content:} \quad I(\text{R-bit}) = -\ln(1 - C_{local})\]

where C_{local} is the coherence between the R-bit and its reference oscillator. Maximum information ($I \to \infty$) corresponds to perfect phase-locking ($C \to 1$).

2.3 R-bit vs. Bit vs. Qubit

Property	Classical Bit	Qubit	R-bit		
States	$\{0, 1\}$	$\alpha	0\rangle + \beta	1\rangle$	$\phi \in [0, 2\pi)$
Information	1 bit	Up to 1 bit (Holevo)	$\log_2(N_{phases}) $		
Error model	Bit-flip	Decoherence	Phase drift		
Correction	Redundancy	Entanglement codes	Phase-locking		
Readout	Voltage threshold	Projective measurement	Phase comparison		
Coupling	Wires	Entanglement	Resonance		

2.4 Multi-Level R-bits

For discrete computation, R-bits can be quantized to N phase levels:

\[\phi_k = \frac{2\pi k}{N}, \quad k = 0, 1, \ldots, N-1\]

A 4-level R-bit stores 2 classical bits. An 8-level R-bit stores 3 bits. The phase-locking dynamics provide natural quantization—the system preferentially locks to one of the N discrete phases.

R-bit Architecture

R-bit Architecture

3. The Coherence Processing Unit (CPU)

3.1 Architecture

A Coherence Processing Unit consists of:

1. **R-bit array:** N coupled oscillators with controllable phases
2. **Lambda-field generator:** A global coherence reference (the "clock")
3. **Coupling matrix:** Programmable inter-oscillator connections
4. **Readout array:** Phase comparators for extracting results

The computation cycle:

1. Initialize R-bit phases (input encoding)
2. Set coupling matrix (program)
3. Allow phase-locking dynamics to evolve
4. Read out final phase configuration (output)

3.2 Programming Model

Instead of gate sequences, a resonant computer is programmed by specifying the **coupling matrix** J_{ij}:

\[\frac{d \phi_i}{dt} = \omega_0 + \sum_j J_{ij} \sin(\phi_j - \phi_i) + \eta \, \nabla \Lambda_i\]

This is the Kuramoto model with Lambda-field coupling. The coupling matrix encodes the "program"—different J_{ij} configurations produce different computations.

3.3 Computation as Phase-Locking

Computation in a resonant computer is the process of phase-locking: the system evolves from an initial phase configuration (input) to a final locked configuration (output). The attractor structure of the dynamics determines the computation:

- **Convergent problems:** All initial conditions flow to the same attractor (unique solution)
- **Branching problems:** Multiple attractors exist (multiple solutions, selected by initial condition)
- **Optimization:** The global minimum of the Lambda-field potential encodes the optimal solution

3.4 Computational Primitives

Operation	Classical Gate	Resonant Equivalent
NOT	Inverter	Phase shift by π
AND	Conjunction	Phase-lock to reference when both inputs aligned
OR	Disjunction	Phase-lock when either input aligned
XOR	Exclusive-or	Phase-lock to difference
Addition	Adder circuit	Frequency beating
Multiplication	Multiplier	Phase modulation
Fourier Transform	FFT algorithm	Natural oscillator spectrum

CPu Architecture

4. Non-Boolean Logic

4.1 Beyond True/False

Boolean logic operates on $\{0, 1\}$. Resonant logic operates on continuous phases, enabling:

- **Fuzzy logic:** Phase proximity represents degree of truth
- **Analog computation:** Continuous-valued operations without discretization
- **Multi-valued logic:** N-level R-bits naturally support N-valued logic
- **Temporal logic:** Phase evolution encodes temporal relationships

4.2 Coherence Logic Gates

Define the coherence between two R-bits:

\[C_{12} = \cos(\phi_1 - \phi_2)\]

Coherence logic operations:

\[\begin{aligned} \text{C-AND}(A, B) &: \quad C_{out} = C_A \cdot C_B && \text{(product of coherences)} \\ \text{C-OR}(A, B) &: \quad C_{out} = 1 - (1 - C_A)(1 - C_B) && \text{(complementary product)} \\ \text{C-NOT}(A) &: \quad C_{out} = 1 - C_A && \text{(coherence complement)} \\ \text{C-IF}(A, B) &: \quad C_{out} = C_A \to C_B && \text{(coherence implication)} \end{aligned}\]

These operations are continuous, differentiable, and naturally composable.

4.3 Pattern Recognition

Resonant computers excel at pattern recognition because phase-locking is inherently a pattern-matching process. Given a stored pattern (a specific phase configuration of the coupling matrix), the system:

1. Receives an input pattern (initial phases)
2. Evolves toward the nearest stored attractor
3. Locks into the matching pattern
4. The match quality is measured by the final coherence C

This is analogous to Hopfield network dynamics but with continuous phases and guaranteed convergence (via the Lambda-field Lyapunov functional).

5. Error Correction

5.1 Natural Error Correction via Phase-Locking

The Lambda-field provides **built-in error correction**. Small phase perturbations (errors) are automatically corrected by the phase-locking dynamics:

\[\frac{d \, \delta\phi}{dt} = -\eta \, \frac{\partial \Lambda}{\partial \phi} \, \delta\phi + \text{noise} \]

For stable fixed points ($\partial^2 \Lambda / \partial \phi^2 < 0$), perturbations decay exponentially with rate:

\[\gamma_{correction} = \eta \, \left| \frac{\partial^2 \Lambda}{\partial \phi^2} \right|\]

5.2 Error Threshold

The error correction succeeds when the correction rate exceeds the noise rate:

\[\gamma_{correction} > \gamma_{noise}\]

This gives a critical coupling strength:

\[\eta_{critical} = \frac{\gamma_{noise}}{\left| \partial^2 \Lambda / \partial \phi^2 \right|}\]

Below $\eta_{critical}$, the R-bit loses phase-lock and errors accumulate.

5.3 Comparison with Quantum Error Correction

Aspect	**Quantum EC**	**Resonant EC**
Mechanism	Entanglement + syndrome	Phase-locking
Overhead	$O(d^2)$ physical qubits per logical	$O(1)$ (built-in)
Threshold	$\sim 10^{-3}$ error rate	$\eta > \eta_{critical}$
Continuous?	Discrete syndromes	Continuous correction
Speed	Discrete correction cycles	Continuous real-time

The key advantage: resonant error correction is continuous and automatic, requiring no discrete correction cycles or ancilla oscillators.

Error Correction

Error Correction

6. Computational Complexity

6.1 What Can Resonant Computers Solve Efficiently?

The computational power of a resonant computer depends on the attractor structure of the phase-locking dynamics. We identify three complexity classes:

1. **R-P (Resonant Polynomial):** Problems solvable by phase-locking in polynomial time

   • Pattern matching, optimization with convex Lambda-landscape
   • Fourier analysis (natural for oscillators)

1. **R-NP (Resonant Non-deterministic Polynomial):** Problems whose solutions can be verified by phase-locking in polynomial time

   • Constraint satisfaction with multiple attractors

1. **R-Hard:** Problems requiring exponential oscillators regardless of coupling

   • Counting problems, arbitrary Boolean circuits

6.2 Natural Speed-ups

Resonant computers provide natural speed-ups for:

Problem Class	Classical	Quantum	Resonant
Fourier Transform	$O(N \log N)$	$O(\log^2 N)$	$O(1)$ (parallel oscillation)
Pattern Match	$O(N)$	$O(\sqrt{N})$	$O(\tau_{lock})$ (constant in N)
Optimization (convex)	$O(N^2)$	$O(N)$	$O(\tau_{lock})$
SAT (general)	$O(2^N)$	$O(2^{N/2})$	$O(2^N)$ (no advantage)

The key insight: resonant computers excel at **continuous optimization** and **pattern recognition** but offer no advantage for discrete combinatorial problems.

6.3 Hybrid Architectures

The optimal architecture combines:

• Classical processors for discrete logic and control
• Resonant processors for optimization and pattern matching
• Quantum processors for problems requiring entanglement (factoring, simulation)

7. Physical Implementations

7.1 Candidate Technologies

Technology	Frequency	Coherence Time	Scalability
Superconducting LC	1-10 GHz	$\sim$ ms	Good (lithography)
Spin-torque oscillators	1-50 GHz	$\sim$ ns	Excellent (CMOS)
Optical parametric	100-400 THz	$\sim$ μs	Moderate (fiber)
MEMS resonators	1-100 MHz	$\sim$ s	Good (MEMS fab)
Josephson junctions	1-20 GHz	$\sim$ ms	Good (existing fab)

7.2 Coupling Mechanisms

The programmable coupling matrix J_{ij} can be implemented via:

- **Electrical:** Capacitive/inductive coupling between LC oscillators
- **Magnetic:** Exchange coupling between spin-torque oscillators
- **Optical:** Nonlinear wave-mixing in parametric oscillators
- **Mechanical:** Strain coupling between MEMS resonators

7.3 The Lambda-Field Generator

The global Lambda-field reference requires a high-coherence master oscillator:

\[C_{master} > 1 - \frac{\epsilon_{target}}{N}\]

where N is the number of R-bits and ϵ_{target} is the acceptable phase error. For $N = 10^6$ R-bits with 10^{-3} phase error, the master must maintain $C_{master} > 0.999999$.

Physical Implementation

Physical Implementation

8. Comparison with Existing Paradigms

8.1 vs. Classical Computing

Aspect	Classical	Resonant
Primitive	Logic gate	Phase-locking

Energy per op	$> k_B T \ln 2$ (Landauer)	$\sim k^*	\Delta \ln C	$
Error model	Bit flips, stuck bits	Phase drift		
Natural problems	Arithmetic, logic	Optimization, matching		
Limitation	Sequential bottleneck	No general-purpose advantage		

8.2 vs. Quantum Computing

Aspect	Quantum	Resonant
Resource	Entanglement	Coherence
Enemy	Decoherence	Phase noise
Error correction	Exponential overhead	Built-in ($O(1)$)
Advantage	Factoring, simulation	Optimization, pattern matching
Temperature	mK (dilution fridge)	Room temperature possible
Readout	Projective (destructive)	Phase comparison (non-destructive)

The most significant advantage of resonant computing over quantum computing is **temperature**: resonant computers can potentially operate at room temperature because phase-locking does not require quantum coherence—it works classically.

8.3 vs. Neuromorphic Computing

Resonant computing shares features with neuromorphic architectures (coupled oscillators, analog computation, pattern matching). The key difference is the Lambda-field: neuromorphic systems rely on learned coupling weights, while resonant computers exploit a global coherence field for error correction and guaranteed convergence.

9. Open Questions

9.1 Universality

Is resonant computing Turing-complete? Preliminary analysis suggests yes (via simulation of Boolean circuits through quantized R-bits), but a formal proof is needed.

9.2 Scalability

The Lambda-field generator sets a fundamental scalability limit. Maintaining phase coherence across N oscillators requires:

$$P_{master} \sim N \cdot \gamma_{noise} \cdot k^{*} T$$

For $N = 10^9$ oscillators at room temperature, this is ~ 1 W—feasible but non-trivial.

9.3 Programming Languages

What does a "programming language" for resonant computers look like? We envision:

- **Declarative:** Specify the desired output pattern (attractor)
- **Constraint-based:** Specify the coupling matrix via optimization constraints
- **Analog:** No discretization of intermediate values

10. Conclusion

Resonant computing represents a fundamentally different approach to computation: instead of manipulating discrete symbols, it manipulates continuous phase relationships. The Lambda-field provides natural error correction, the Kuramoto dynamics provide computation, and phase-locking provides output. While not a universal replacement for classical or quantum computing, resonant computers offer significant advantages for optimization, pattern recognition, and analog signal processing—precisely the domains where biological neural systems excel.

The resonant computer is not a faster classical computer. It is a different kind of computer—one that computes the way nature does.

Connections

- [[FRC-100-001]] — Coherence definition (the fundamental quantity computed)
- [[FRC-100-007]] — Lambda-field dynamics (the error correction mechanism)
- [[FRC-566-001]] — UCC as the governing equation for R-bit dynamics
- [[FRC-100-006]] — Born rule and attractor basins (computational complexity)

References

1. Kuramoto, Y. (1984). Chemical oscillations, waves, and turbulence. Springer.
2. Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. PNAS, 79(8), 2554-2558.
3. Csaba, G., & Porod, W. (2020). Coupled oscillators for computing: A review and perspective. Applied Physics Reviews, 7(1), 011302.
4. Vodenicarevic, D., et al. (2017). Low-energy truly random number generation with superparamagnetic tunnel junctions for unconventional computing. Physical Review Applied, 8(5), 054045.

5. Landauer, R. (1961). Irreversibility and heat generation in the computing process. IBM Journal of Research and Development, 5(3), 183-191.
6. Acebrón, J. A., et al. (2005). The Kuramoto model: A simple paradigm for synchronization phenomena. Reviews of Modern Physics, 77(1), 137.