

# Standard Operating Procedure

---

**Document Number:** RING-SYS-001

**Standard Operating Procedure (SOP):** Ring Oscillator Logic System

**Revision:** 1.0

**Last Updated:** [DATE]

---

## Contents

---

1. Introduction
  - 1.1 Purpose
  - 1.2 System Overview
  - 1.3 Regulatory Compliance
2. System Specifications
  - 2.1 Logical Parameters and Variables
  - 2.2 Hardware Configuration
3. Operational Protocols
  - 3.1 Initialization Sequence
  - 3.2 Ring Oscillation Cycle
  - 3.3 Inverter Behavior
4. Emergency Operations
  - 4.1 Oscillation Fault Detection
  - 4.2 Fail-Safe Logic Response
5. Maintenance Requirements
  - 5.1 Signal Integrity Checks
  - 5.2 Gate Recalibration
6. Quality Assurance
  - 6.1 Logical Invariance Verification
  - 6.2 Liveness Property Conformance

- 7. Security Protocols
    - 7.1 State Stability Enforcement
    - 7.2 Redundancy and Verification
  - 8. Environmental Considerations
    - 8.1 Electrical Noise Handling
    - 8.2 Temperature Stability
  - 9. Training Requirements
    - 9.1 System Behavior Interpretation
    - 9.2 Model Checking Tools Familiarity
  - 10. Document Control
    - 10.1 Revision History
    - 10.2 Authorization
  - 11. Process Flows and State Transitions
    - 11.1 Inverter State Transitions
    - 11.2 Oscillation Feedback Loop
    - 11.3 Fairness and Liveness Conditions
- 

# **1. Introduction**

---

## **1.1 Purpose**

---

- Define safe and predictable behavior for a 3-inverter ring oscillator system.
- Provide operational guidance and verification protocols for oscillation logic.

## **1.2 System Overview**

---

- Comprises three logic inverters connected in a closed loop.
- Output of each inverter is the input to the next, creating continuous inversion.
- System exhibits oscillatory behavior based on logical negation across the ring.

### 1.3 Regulatory Compliance

---

- Compliant with formal verification principles for liveness and safety.
  - Aligned with logical modeling practices for real-time embedded systems.
- 

## 2. System Specifications

---

### 2.1 Logical Parameters and Variables

---

- `output` : Boolean value indicating logical HIGH ( `TRUE` ) or LOW ( `FALSE` ).
- `input` : Signal received from previous inverter in the ring.
- Initial state: `FALSE` for all outputs.
- Transition logic: `next(output) := !input` .

### 2.2 Hardware Configuration

---

- Three inverter modules: `gate1`, `gate2`, `gate3`.
  - Connectivity:
    - `gate1.input`  $\leftarrow$  `gate3.output`
    - `gate2.input`  $\leftarrow$  `gate1.output`
    - `gate3.input`  $\leftarrow$  `gate2.output`
- 

## 3. Operational Protocols

---

### 3.1 Initialization Sequence

---

- On startup, all inverter outputs are initialized to `FALSE` .
- State machine transitions begin immediately via logical negation.

- Fair scheduling assumed for each module ( `FAIRNESS` running ).

## 3.2 Ring Oscillation Cycle

---

- The three inverters alternate states in sequence:
  - `gate1` → `gate2` → `gate3` → `gate1` ...
- Oscillation relies on asynchronous signal transitions and inversion.
- No external clock is required; behavior is self-timed.

## 3.3 Inverter Behavior

---

- Each inverter updates its output based on the inverse of its input.
  - Outputs toggle as the system progresses through its ring cycle.
  - Ensures continuous state change as long as the process is fair.
- 

# 4. Emergency Operations

---

## 4.1 Oscillation Fault Detection

---

- Monitor for frozen state (no toggling over time).
- If `AG AF gate1.output` or `AG AF !gate1.output` fails, raise fault alert.

## 4.2 Fail-Safe Logic Response

---

- Halt oscillation propagation if one inverter becomes unresponsive.
  - Reset output values or trigger manual reinitialization.
-

## 5. Maintenance Requirements

---

### 5.1 Signal Integrity Checks

---

- Verify each inverter receives valid input from preceding gate.
- Check for transient faults that may introduce static loops.

### 5.2 Gate Recalibration

---

- Inspect inverter thresholds periodically.
- Replace components showing drift in logical thresholds or toggling failure.

---

## 6. Quality Assurance

---

### 6.1 Logical Invariance Verification

---

- Ensure that the output of any gate will eventually toggle:
  - `AG AF gate1.output` and `AG AF !gate1.output` hold true.

### 6.2 Liveness Property Conformance

---

- Confirm that the system does not settle into a stable state.
  - Use model checkers to verify alternating truth values over infinite runs.
-

## 7. Security Protocols

---

### 7.1 State Stability Enforcement

---

- Guard against indefinite holding of a single logical value.
- Introduce watchdog logic if necessary to ensure toggling.

### 7.2 Redundancy and Verification

---

- Redundant gates may be added for error detection.
- Simulation and formal verification tools must confirm correct transitions.

---

## 8. Environmental Considerations

---

### 8.1 Electrical Noise Handling

---

- Include debouncing or filtering for voltage fluctuations.
- Shield wiring between gates in physical implementations.

### 8.2 Temperature Stability

---

- Validate inverter thresholds across temperature ranges.
- Incorporate temperature-compensated logic if needed.

---

## 9. Training Requirements

---

### 9.1 System Behavior Interpretation

---

- Operators must understand ring oscillator behavior and state toggling.

- Training on interpreting logical waveforms over time.

## 9.2 Model Checking Tools Familiarity

---

- Team must be trained on using formal verification tools (e.g., NuSMV).
  - Interpret results of safety ( `AG` ) and liveness ( `AF` ) specifications.
- 

# 10. Document Control

---

## 10.1 Revision History

---

- Rev 1.0 - Initial SOP derived from formal ring oscillator specification (ring.txt)

## 10.2 Authorization

---

- Approved by: Hardware Logic Design Lead
  - Reviewed by: Verification and Validation Team
- 

# 11. Process Flows and State Transitions

---

## 11.1 Inverter State Transitions

---

- State at time `t+1`: `output = NOT(input)`
- Inputs are derived from the output of the previous inverter in the ring.

## 11.2 Oscillation Feedback Loop

---

- `gate1.output` → `gate2.input`
- `gate2.output` → `gate3.input`
- `gate3.output` → `gate1.input`

- Loop ensures feedback and propagation of toggling behavior.

### 11.3 Fairness and Liveness Conditions

---

- The `FAIRNESS running` constraint ensures each inverter will execute.
- Liveness properties:
  - Eventually gate1 will be `TRUE`
  - Eventually gate1 will be `FALSE`