Monitoring Cyber-Physical Systems: From Design to Integration

Maximilian Schwenger
OVERVIEW

BLOCK I
- TYPE-CHECK
- INTERPRETATION
- SPECIFICATION

BLOCK II
- HARDWARE
- SOFTWARE

BLOCK III
- POST-MORTEM
- INTEGRATION
1. Never injure humans.
2. Obey orders.
3. Protect yourself.

**Our Setup**

**System**

**Monitor**

**Health**

**Analysis**

**SPEC**
### Property Spectrum

<table>
<thead>
<tr>
<th>Sensor Level</th>
<th>Mission Level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timeliness</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Arithmetic Challenge</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Input Data</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Locality</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td></td>
</tr>
</tbody>
</table>
# Property Spectrum

<table>
<thead>
<tr>
<th>Sensor Level</th>
<th>Mission Level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timeliness</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Arithmetic Challenge</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Input Data</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Locality</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><strong>Mission Statistics:</strong></td>
</tr>
<tr>
<td>Data Validation: “Altimeter must produce positives values.”</td>
<td>“Low correlation between WP distance and relative path deviation.”</td>
</tr>
</tbody>
</table>
## Property Spectrum

<table>
<thead>
<tr>
<th>Sensor Level</th>
<th>Mission Level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timeliness</strong></td>
<td>Critical</td>
</tr>
<tr>
<td><strong>Arithmetic Challenge</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Input Data</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Locality</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Data Validation: “Altimeter must produce positive values.”</td>
</tr>
</tbody>
</table>
### Block I: Property Spectrum

<table>
<thead>
<tr>
<th>Sensor Level</th>
<th>Mission Level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timeliness</strong></td>
<td>Critical</td>
</tr>
<tr>
<td><strong>Arithmetic Challenge</strong></td>
<td>Low</td>
</tr>
<tr>
<td>(bounds checks, counting)</td>
<td></td>
</tr>
<tr>
<td><strong>Input Data</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Locality</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Data Validation:</td>
</tr>
<tr>
<td></td>
<td>“Altimeter must produce positives values.”</td>
</tr>
<tr>
<td></td>
<td>Mission Statistics:</td>
</tr>
<tr>
<td></td>
<td>“Low correlation between WP distance and relative path deviation.”</td>
</tr>
<tr>
<td>Sensor Level</td>
<td>Mission Level</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td><strong>Timeliness</strong></td>
<td><strong>Critical</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Lax</strong></td>
</tr>
<tr>
<td><strong>Arithmetic</strong></td>
<td><strong>Low</strong></td>
</tr>
<tr>
<td><strong>Challenge</strong></td>
<td><strong>High</strong></td>
</tr>
<tr>
<td>(bounds checks, counting)</td>
<td>(statistics, prediction)</td>
</tr>
<tr>
<td><strong>Input Data</strong></td>
<td><strong>Raw</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Refined</strong></td>
</tr>
<tr>
<td><strong>Locality</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><strong>Data Validation:</strong></td>
</tr>
<tr>
<td></td>
<td>“Altimeter must produce positives values.”</td>
</tr>
<tr>
<td></td>
<td><strong>Mission Statistics:</strong></td>
</tr>
<tr>
<td></td>
<td>“Low correlation between WP distance and relative path deviation.”</td>
</tr>
</tbody>
</table>
## Property Spectrum

<table>
<thead>
<tr>
<th>Sensor Level</th>
<th>Mission Level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timeliness</strong></td>
<td><strong>Critical</strong></td>
</tr>
<tr>
<td><strong>Arithmetic Challenge</strong></td>
<td><strong>Low</strong></td>
</tr>
<tr>
<td></td>
<td><em>(bounds checks, counting)</em></td>
</tr>
<tr>
<td><strong>Input Data</strong></td>
<td><strong>Raw</strong></td>
</tr>
<tr>
<td><strong>Locality</strong></td>
<td><strong>Local</strong></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><strong>Data Validation:</strong></td>
</tr>
<tr>
<td></td>
<td><em>“Altimeter must produce positives values.”</em></td>
</tr>
</tbody>
</table>
Block I

One Language to Rule Them All?
Block I

One Language to Rule Them All?

Control + Guarantees

Logics

Expressiveness

General-Purpose Languages
Points to consider:

- Output Quality
- Expressiveness
- Integration
- Guarantees
- Certifiability

One Language to Rule Them All?

Control + Guarantees

Expressiveness

Logics

General-Purpose Languages
Points to consider:
❖ Output Quality → Stream-based + Quantitative
❖ Expressiveness
❖ Integration
❖ Guarantees
❖ Certifiability
Points to consider:

❖ Output Quality → Stream-based + Quantitative
❖ Expressiveness → Arithmetic + Clarity
❖ Integration
❖ Guarantees
❖ Certifiability
Block I

One Language to Rule Them All?

Points to consider:

❖ Output Quality → Stream-based + Quantitative
❖ Expressiveness → Arithmetic + Clarity
❖ Integration → HW + SW compilation
❖ Guarantees
❖ Certifiability

Control + Guarantees

Expressiveness

Logics

General-Purpose Languages
Points to consider:

❖ Output Quality → Stream-based + Quantitative
❖ Expressiveness → Arithmetic + Clarity
❖ Integration → HW + SW compilation
❖ Guarantees → Const Space + Const Time per event
❖ Certifiability
Points to consider:

❖ Output Quality → Stream-based + Quantitative
❖ Expressiveness → Arithmetic + Clarity
❖ Integration → HW + SW compilation
❖ Guarantees → Const Space + Const Time per event
❖ Certifiability → SW verified; prelim results for HW
Sensor Validation 1: The altimeter readings must be non-negative.

Input altitude: Float32
Trigger altitude < 0 “Altimeter reports negative values.”
Sensor Validation 1: *Altimeter readings must be non-negative.*

- **Input:** altitude: Float32
- **Trigger:** altitude < 0 “Altimeter reports negative values.”

Sensor Validation 2: *Barometer must produce 9 — 11 readings per second.*

- **Input:** pressure: Float32
- **Output:** read_ps @ 1Hz := pressure.aggregate(over: 1s, using: count)
- **Trigger:** read_ps > 11 ∨ read_ps < 9 “Barometer count irregular.”
Sensor Validation 1: *Altimeter readings must be non-negative.*

- **input altitude:** Float32
- **trigger altitude < 0** “Altimeter reports negative values.”

Sensor Validation 2: *Barometer must produce 9 — 11 readings per second.*

- **input pressure:** Float32
- **output read_ps @ 1Hz** := pressure.aggregate(over: 1s, using: count)
- **trigger read_ps > 11 ∨ read_ps < 9** “Barometer count irregular.”
Sensor Validation 1: Altimeter readings must be non-negative.

- input altitude: Float32
- trigger altitude < 0 “Altimeter reports negative values.”

Sensor Validation 2: Barometer must produce 9 — 11 readings per second.

- input pressure: Float32
- output read_ps @ 1Hz := pressure.aggregate(over: 1s, using: count)
- trigger read_ps > 11 ∨ read_ps < 9 “Barometer count irregular.”
Sensor Validation 1: Altimeter readings must be non-negative.

Sensor Validation 2: Barometer must produce 9 – 11 readings per second.

Input pressure: Float32

Output read_ps @ 1Hz := pressure.aggregate(over: 1s, using: count)

Trigger read_ps > 11 ∨ read_ps < 9 “Barometer count irregular.”
Sensor Validation 1: Altimeter readings must be non-negative.

Trigger altitude < 0 “Altimeter reports negative values.”

Sensor Validation 2: Barometer must produce 9 – 11 readings per second.

Input pressure: Float32
Output read_ps @ 1Hz := pressure.aggregate(over: 1s, using: count)
Trigger read_ps > 11 ∨ read_ps < 9 “Barometer count irregular.”
Sensor Validation 1: Altimeter readings must be non-negative.

Sensor Validation 2: Barometer must produce 4 readings per second.

input pressure: Float32
output read_ps @ 1Hz := pressure.aggregate(over: 1s, using: count)
trigger read_ps > 11 ∨ read_ps < 9 “Barometer count irregular.”
Every request needs to be granted within a second.

MTL: $\Box (r \rightarrow \Diamond_{1s} g)$
Every request needs to be granted within a second.

MTL: $\Box (r \rightarrow \Diamond_{1s} g)$
Every request needs to be granted within a second.

MTL: $\square (r \rightarrow \Diamond_{1s} g)$
Every request needs to be granted within a second.

MTL: \( \Box (r \rightarrow \Diamond_{1s} g) \)
Every request needs to be granted within a second.

MTL: $\square (r \rightarrow \Diamond_{1s} g)$
Every request needs to be granted within a second.

\[
\text{MTL: } \square (r \rightarrow \diamond_{1s} g)
\]
**Block I**

**RTLola’s Type System I**

### Value Type

- Any
- Bool
- Numeric
  - Float64
  - Integer
    - Float32
    - Int64
    - Unsigned
      - Int32
      - UInt64

### Pacing Type

- Any
- Periodic
  - 6Hz
  - 3Hz
  - 2Hz
  - 1Hz
- Event
  - \(a \lor b\)
  - \(a \land b\)
**Block I**

**2-Dimensional Time**

- **$e_1$: event**
- **$p_1$: periodic**
- **$p_2$: periodic**
- **$e_2$: event**

- **0-order hold**

Time points:
- 1s
- 2s
- 3s
- 4s
Synchronous: Default; Couples timing; Infallible
Holds + aggregations: Decouple timing; Fallible
Aggregations only permitted in periodic streams
Sensor Validation 1: *Altimeter readings must be non-negative.*

input altitude: Float32

trigger altitude < 0 “Altimeter reports negative values.”
Sensor Validation 1: Altimeter readings must be non-negative.

input altitude: Float32 Float32 | altitude

trigger altitude < 0 “Altimeter reports negative values.”
Sensor Validation 1: *Altimeter readings must be non-negative.*

- input altitude: Float32
  - Float32 | altitude
- trigger altitude < 0 “Altimeter reports negative values.”
  - Bool | altitude
Sensor Validation 1: *Altimeter readings must be non-negative.*

input altitude: Float32\[\text{Float32}|\text{altitude}\]

trigger altitude < 0 “Altimeter reports negative values.”

Ensures timeliness
Sensor Validation 1: *Altimeter readings must be non-negative.*

input altitude: Float32

trigger altitude < 0 “Altimeter reports negative values.”

Sensor Validation 2: *Barometer must produce 9 — 11 readings per second.*

input pressure: Float32

output read_ps @ 1Hz := pressure.aggregate(over: 1s, using: count)

trigger read_ps > 11 ∨ read_ps < 9 “Barometer count irregular.”
Sensor Validation 1: *Altimeter readings must be non-negative.*

\[
\text{input altitude: Float32} \quad \text{Float32} \mid \text{altitude} \\
\text{trigger altitude} < 0 \quad \text{“Altimeter reports negative values.”} \quad \text{Bool} \mid \text{altitude}
\]

Sensor Validation 2: *Barometer must produce 9 — 11 readings per second.*

\[
\text{input pressure: Float32} \quad \text{Float32} \mid \text{pressure} \\
\text{output read_ps} \at 1\text{Hz} := \text{pressure.aggregate(over: 1s, using: count)} \\
\text{trigger read_ps} > 11 \lor \text{read_ps} < 9 \quad \text{“Barometer count irregular.”}
\]
Sensor Validation 1: *Altimeter readings must be non-negative.*

- **Input:** 
  - `input altitude: Float32` 
  - `Float32 | altitude`

- **Trigger:**
  - `trigger altitude < 0 "Altimeter reports negative values."` 
  - `Bool | altitude`

Sensor Validation 2: *Barometer must produce 9 — 11 readings per second.*

- **Input:** 
  - `input pressure: Float32` 
  - `Float32 | pressure`
  - `UInt32 | 1Hz`

- **Output:**
  - `output read_ps @ 1Hz := pressure.aggregate(over: 1s, using: count)`

- **Trigger:**
  - `trigger read_ps > 11 ∨ read_ps < 9 "Barometer count irregular."`
Sensor Validation 1: *Altimeter readings must be non-negative.*

```
input altitude: Float32 Float32 | altitude
trigger altitude < 0 "Altimeter reports negative values." Bool | altitude
```

Sensor Validation 2: *Barometer must produce 9 — 11 readings per second.*

```
input pressure: Float32 Float32 | pressure UInt32 | 1Hz
output read_ps @ 1Hz := pressure.aggregate(over: 1s, using: count)
trigger read_ps > 11 v read_ps < 9 "Barometer count irregular." Bool | 1Hz
```
Sensor Validation 1: *Altimeter readings must be non-negative.*

input altitude: Float32 \(\text{Float32} \mid \text{altitude}\)

trigger altitude < 0 “Altimeter reports negative values.” \(\text{Bool} \mid \text{altitude}\)

Sensor Validation 2: *Barometer must produce 9 — 11 readings per second.*

input pressure: Float32 \(\text{Float32} \mid \text{pressure}\) UInt32 \(\text{ UInt32} \mid \text{1Hz}\)

output read_ps @ 1Hz := pressure.aggregate(over: 1s, using: count)

trigger read_ps > 11 \(\lor\) read_ps < 9 “Barometer count irregular.” \(\text{Bool} \mid \text{1Hz}\)

output x := pressure.aggregate(…)

output y := read_ps * pressure
Sensor Validation 1: *Altimeter readings must be non-negative.*

\[
\text{input altitude: Float32 } \quad \text{Float32 | altitude}
\]

\[
\text{trigger altitude < 0 “Altimeter reports negative values.” } \quad \text{Bool | altitude}
\]

Sensor Validation 2: *Barometer must produce 9 — 11 readings per second.*

\[
\text{input pressure: Float32 } \quad \text{Float32 | pressure } \quad \text{UInt32 | 1Hz}
\]

\[
\text{output read_ps @ 1Hz := pressure.aggregate(over: 1s, using: count)}
\]

\[
\text{trigger read_ps > 11 } \lor \text{ read_ps < 9 “Barometer count irregular.” } \quad \text{Bool | 1Hz}
\]

\[
\text{output x := pressure.aggregate(...) aggregation w/o period}
\]

\[
\text{output y := read_ps } \ast \text{ pressure}
\]
Sensor Validation 1: *Altimeter readings must be non-negative.*

```plaintext
input altitude: Float32  Float32 | altitude
trigger altitude < 0 “Altimeter reports negative values.”  Bool | altitude
```

Sensor Validation 2: *Barometer must produce 9 — 11 readings per second.*

```plaintext
input pressure: Float32  Float32 | pressure  UInt32 | 1Hz
output read_ps @ 1Hz := pressure.aggregate(over: 1s, using: count)
trigger read_ps > 11 ∨ read_ps < 9 “Barometer count irregular.”  Bool | 1Hz

output x := pressure.aggregate(...)  aggregation w/o period
output y := read_ps * pressure  mixes periodic and events
```
input velo_1, velo_2: Int32

output deviation := abs(velo_1 - velo_2)
Int32 | velo_1  Int32 | velo_2

input velo_1, velo_2: Int32

UInt32 | velo_1 ∧ velo_2

output deviation  @ velo_1 ∧ velo_2 := abs(velo_1 - velo_2)
**Block I  **

**RTLola’s Type System III**

\[ \text{Int32} \mid \text{velo}_1 \quad \text{Int32} \mid \text{velo}_2 \]

**Input**

\[
\text{velo}_1, \text{velo}_2 : \text{Int32}
\]

**Output**

\[ \text{output deviation } @ \text{velo}_1 \land \text{velo}_2 := \text{abs(velo}_1 - \text{velo}_2) \]

\[ \text{output deviation’ } @ \text{velo}_1 \lor \text{velo}_2 \]

\[ := \text{abs(velo}_1 \lor \text{velo}_2 - \text{velo}_2) \]
Int32 | velo_1    Int32 | velo_2

input velo_1, velo_2: Int32

UInt32 | velo_1 ∧ velo_2

output deviation @ velo_1 ∧ velo_2 := abs(velo_1 - velo_2)

UInt32 | velo_1 ∨ velo_2

output deviation’ @ velo_1 ∨ velo_2
    := abs(velo_1.hold(or: 0) - velo_2.hold(or: 0))
**Block I**  
**RTLola’s Type System III**

Int32 | velo_1   Int32 | velo_2

input velo_1, velo_2: Int32

UInt32 | velo_1 ∧ velo_2

output deviation @ velo_1 ∧ velo_2 := abs(velo_1 - velo_2)

UInt32 | velo_1 ∨ velo_2

output deviation’ @ velo_1 ∨ velo_2
:= abs(velo_1.\hold\(or: 0\) - velo_2.\hold\(or: 0\))
Block I

**RTLola’s Type System III**

Int32 | velo_1  Int32 | velo_2

input velo_1, velo_2: Int32

UInt32 | velo_1 ∧ velo_2

output deviation @ velo_1 ∧ velo_2 := abs(velo_1 - velo_2)

UInt32 | velo_1 ∨ velo_2

output deviation’ @ velo_1 ∨ velo_2

:= abs(velo_1.hold(or: 0) - velo_2.hold(or: 0))

Int32 | Any

Int32 | Any

**Strong type system supports specifier.**

→ Increases confidence in spec.
Mission Statistic: Does the WP-distance correlate with the relative path deviation?

input wp, pos: (Float64, Float64)
Mission Statistic: *Does the WP-distance correlate with the relative path deviation?*

input \( wp, pos: (\text{Float64}, \text{Float64}) \)

output \( wp\_dist := \text{abs}(wp - wp.\text{offset(by: -1, dft: wp})) \)
**Mission Statistic:** Does the WP-distance correlate with the relative path deviation?

- **Input:** \(wp, \text{pos}: \text{(Float64, Float64)}\)
- **Output:**
  - \(wp\_dist := \text{abs}(wp - wp.\text{offset(by: -1, dft: wp)})\)
  - \(\text{dist\_total} := \text{pos} - \text{pos.offset(by: -1, dft: START)} + \text{dist\_total.offset(by: -1, dft: 0)}\)
Mission Statistic: Does the WP-distance correlate with the relative path deviation?

input wp, pos: (Float64, Float64)

output wp_dist := abs(wp - wp.offset(by: -1, dft: wp))
output dist_total := pos - pos.offset(by: -1, dft: START)
                          + dist_total.offset(by: -1, dft: 0)
output total_dist_at_wp @ wp := dist_total.hold(or: 0)

output devi := abs(wp_dist.offset(by: -1, dft: 0) - (total_dist_at_wp - total_dist_at_wp.offset(by: -1, dft: 0))
output dist_v_devi := (wp_dist, devi)
output cov := dist_v_devi.aggregate(over: ∞, using: cov)
output var_dist := wp_dist.aggregate(over: ∞, using: var)
output var_devi := devi.aggregate(over: ∞, using: var)
output corr := cov / (var_devi^2 * var_dist^2)
Mission Statistic: Does the WP-distance correlate with the relative path deviation?

input wp, pos: (Float64, Float64)

output wp_dist := abs(wp - wp.offset(by: -1, dft: wp))
output dist_total := pos - pos.offset(by: -1, dft: START)
    + dist_total.offset(by: -1, dft: 0)
output total_dist_at_wp @ wp := dist_total.hold(or: 0)
output devi @ wp := abs( wp_dist.offset(by: -1, dft: 0) -
    (total_dist_at_wp - total_dist_at_wp.offset(by: -1, dft: 0) ) )
Mission Statistic: Does the WP-distance correlate with the relative path deviation?

input wp, pos: (Float64, Float64)

output wp_dist := abs(wp - wp.offset(by: -1, dft: wp))
output dist_total := pos - pos.offset(by: -1, dft: START)
    + dist_total.offset(by: -1, dft: 0)
output total_dist_at_wp @ wp := dist_total.hold(or: 0)
output devi @ wp := abs(wp_dist.offset(by: -1, dft: 0) -
    (total_dist_at_wp - total_dist_at_wp.offset(by: -1, dft: 0))

output dist_v_devi @ wp := (wp_dist, devi)
output cov @ 1Hz := dist_v_devi.aggregate(over: ∞, using: cov)
output var_dist @ 1Hz := wp_dist.aggregate(over: ∞, using: var)
output var_devi @ 1Hz := devi.aggregate(over: ∞, using: var)
output corr := cov / (var_devi^2 * var_dist^2)
Mission Statistic: Does the WP-distance correlate with the relative path deviation?

input \( wp, \ pos: (\text{Float64}, \text{Float64}) \)

output \( wp\_dist := \text{abs}(wp - wp.\text{offset}(by: -1, \text{dft: wp})) \)
output \( \text{dist}\_\text{total} := \text{pos} - \text{pos.}\text{offset}(by: -1, \text{dft: START}) \)
output \( \text{total}\_\text{dist}\_\text{at}\_wp @ wp := \text{dist}\_\text{total.}\text{hold(or: 0)} \)
output \( \text{devi} @ wp := \text{abs}(wp\_\text{dist.}\text{offset}(by: -1, \text{dft: 0}) - (\text{total}\_\text{dist}\_\text{at}\_wp - \text{total}\_\text{dist}\_\text{at}\_wp.\text{offset}(by: -1, \text{dft: 0}) ) \)
output \( \text{dist}\_\text{v}\_\text{devi} @ wp := (wp\_dist, \text{devi}) \)

\[ \text{cov} \_1\text{Hz} := \text{dist}\_\text{v}\_\text{devi.}\text{aggregate(over: }\infty\text{, using: }\text{cov}) \]
output \( \text{var}\_\text{dist} \_1\text{Hz} := \text{wp}\_\text{dist.}\text{aggregate(over: }\infty\text{, using: }\text{var}) \)
output \( \text{var}\_\text{devi} \_1\text{Hz} := \text{devi.}\text{aggregate(over: }\infty\text{, using: }\text{var}) \)
output \( \text{corr} := \text{cov} / (\text{var}\_\text{devi}\_1\text{Hz}^2 \ast \text{var}\_\text{dist}\_1\text{Hz}^2) \)

**RTLola provides primitives for abstract, mission-level properties.**
Great, we’ve got a spec and the type checker is happy. What next?

A) Further increase confidence  B) Analyze complexity
Block I

Validation via Interpretation
**SPECIFICATION:**

- GPS frequency validation
- GPS/IMU jump detection
- Hover phase detection

**RESULTS:**

- 433,000 events
- 1,545ns per event @ 146%
- Stack size < 1kB, no heap
**SPECIFICATION:**

GPS frequency validation
GPS/IMU jump detection
Hover phase detection

**RESULTS:**

433,000 events
1,545ns per event @ 146%
Stack size < 1kB, no heap

ENABLES RAPID DEVELOPMENT.
Block I  Specification Analysis
<table>
<thead>
<tr>
<th>Stream</th>
<th>#values</th>
<th>Size</th>
<th>Windows</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>pos</td>
<td>2</td>
<td>128</td>
<td></td>
<td>256</td>
</tr>
<tr>
<td>wp</td>
<td>2</td>
<td>128</td>
<td></td>
<td>256</td>
</tr>
<tr>
<td>d_total</td>
<td>2</td>
<td>64</td>
<td></td>
<td>128</td>
</tr>
<tr>
<td>wp_dist</td>
<td>1</td>
<td>64</td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>d_s_wp</td>
<td>1</td>
<td>64</td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>devi</td>
<td>1</td>
<td>64</td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>var_dist</td>
<td>1</td>
<td>64</td>
<td>128</td>
<td>192</td>
</tr>
<tr>
<td>di_v_de</td>
<td>1</td>
<td>64</td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>var_dist</td>
<td>1</td>
<td>64</td>
<td>128</td>
<td>192</td>
</tr>
<tr>
<td>cov</td>
<td>1</td>
<td>64</td>
<td>128</td>
<td>192</td>
</tr>
<tr>
<td>corr</td>
<td>1</td>
<td>64</td>
<td></td>
<td>64</td>
</tr>
</tbody>
</table>
## Block I Specification Analysis

<table>
<thead>
<tr>
<th>Stream</th>
<th>#values</th>
<th>Size</th>
<th>Windows</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>pos</td>
<td>2</td>
<td>128</td>
<td>256</td>
<td>128</td>
</tr>
<tr>
<td>wp</td>
<td>2</td>
<td>128</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>d_total</td>
<td>2</td>
<td>64</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>wp_dist</td>
<td>1</td>
<td>64</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>d_s_wp</td>
<td>1</td>
<td>64</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>devi</td>
<td>1</td>
<td>64</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>var_dist</td>
<td>1</td>
<td>64</td>
<td>128</td>
<td>192</td>
</tr>
<tr>
<td>di_v_de</td>
<td>1</td>
<td>64</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>var_dist</td>
<td>1</td>
<td>64</td>
<td>128</td>
<td>192</td>
</tr>
<tr>
<td>cov</td>
<td>1</td>
<td>64</td>
<td>128</td>
<td>192</td>
</tr>
<tr>
<td>corr</td>
<td>1</td>
<td>64</td>
<td>128</td>
<td>64</td>
</tr>
</tbody>
</table>

\[ \Sigma 1536B \]
<table>
<thead>
<tr>
<th>Sensor Level</th>
<th>Mission Level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timeliness</strong></td>
<td><strong>Critical</strong></td>
</tr>
<tr>
<td><strong>Arithmetic Challenge</strong></td>
<td><strong>Low</strong></td>
</tr>
<tr>
<td></td>
<td><em>(bounds checks, counting)</em></td>
</tr>
<tr>
<td><strong>Input Data</strong></td>
<td><strong>Raw</strong></td>
</tr>
<tr>
<td><strong>Locality</strong></td>
<td><strong>Local</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Inter-component</strong></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Data Validation:</td>
</tr>
<tr>
<td></td>
<td>“Altimeter must produce</td>
</tr>
<tr>
<td></td>
<td>positives values.”</td>
</tr>
<tr>
<td></td>
<td>Mission Statistics:</td>
</tr>
<tr>
<td></td>
<td>“Low correlation between WP</td>
</tr>
<tr>
<td></td>
<td>distance and relative path</td>
</tr>
<tr>
<td></td>
<td>deviation.”</td>
</tr>
</tbody>
</table>
Every request needs to be granted within a second.

MTL: □(r → ◇_1s g)

Check t = 1.93

MTL:  

RTLola: Check t = 1.07
Every request needs to be granted within a second.
Every request needs to be granted within a second.

**SPECIFICATION:**
- GPS frequency validation
- GPS/IMU jump detection
- Hover phase detection

**RESULTS:**
- 433,000 events
- 1,545ns per event @ 146%
- Stack size < 1kB, no heap
Every request needs to be granted within a second.

<table>
<thead>
<tr>
<th>Stream</th>
<th>#values</th>
<th>Size</th>
<th>Windows</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>pos</td>
<td>2</td>
<td>128</td>
<td></td>
<td>256</td>
</tr>
<tr>
<td>wp</td>
<td>2</td>
<td>128</td>
<td></td>
<td>256</td>
</tr>
<tr>
<td>d_total</td>
<td>2</td>
<td>64</td>
<td></td>
<td>128</td>
</tr>
<tr>
<td>wp_dist</td>
<td>1</td>
<td>64</td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>d_s_wp</td>
<td>1</td>
<td>64</td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>devi</td>
<td>1</td>
<td>64</td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>var_dist</td>
<td>1</td>
<td>64</td>
<td>128</td>
<td>192</td>
</tr>
<tr>
<td>di_v_de</td>
<td>1</td>
<td>64</td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>var_dist</td>
<td>1</td>
<td>64</td>
<td>128</td>
<td>192</td>
</tr>
<tr>
<td>cov</td>
<td>1</td>
<td>64</td>
<td>128</td>
<td>192</td>
</tr>
<tr>
<td>corr</td>
<td>1</td>
<td>64</td>
<td></td>
<td>64</td>
</tr>
</tbody>
</table>

Σ 1536B
**Block II Overview**

**Block I**
- Type-Check
- Interpretation
- Specification

**Block II**
- Hardware
- Software

**Block III**
- Post-Mortem
- Integration
Block II

Interpretation v Compilation

Specification

Never injure humans. Obey orders. Protect yourself.

Interprets

Observes

Monitor

01010010
01010110
00110010
00110000
00110000
**Block II**

**Interpretation v Compilation**

Specification:

Never injure humans. Obey orders. Protect yourself.

Compilation:

```rust
impl Monitor {
    while let Some(i) = get_input() {
        ...  
    }
}
```

High Level Code:

```
01010010
01010110
00110010
00110000
```

Monitor Observes:

```
01010010
01010110
00110010
00110000
```

20
Specification

Never injure humans. Obey orders. Protect yourself.

Compilation

Impl Monitor {
  while let Some(i) = get_input() {
    ...
  }
}

VHDL Code

01010010
01010110
00110010
00110000
01010010
01010110
00110010
00110000

MONITOR

Observes

Block II Interpretation v Compilation
**Block II**  
**CHALLENGES**

- Reduce Circuit Cost
- Utilize Parallel Execution
- Periodic versus Event-Based
1. Never injure humans.
2. Obey orders.
3. Protect yourself.

Challenges:

Block II: VHDL Compilation Overview
Challenges:

- Reception
- Q-Conn
- QUEUE
- HLC
- LLC
- idle
- pop
- eval
- λ

Input 1, Input n, Output λ₁, Output λ₂, Output λ₃, Output λ₄, Output λ₅, 1, 2.0, 2.λ
**Block II**  
**VHDL Compilation Overview**

**HLC**
- **Reception**
- **Q-Conn**

**QUEUE**

**LLC**
- **idle**
- **pop**
- **eval**
- **2.0**
- **2.λ**

**Challenges:**
- Input 1
- Input n
- Output λ₁,₁
- Output λ₁,n
- Output λ₂,₁
- Output λ₂,n
Block II VHDL Compilation Overview

- Reception
- Q-Conn
- QUEUE

HLC

LLC

- idle
- pop
- eval
- idle

Flow:
- Input 1
- Input n
- Output $\lambda_{1,1}$
- Output $\lambda_{1,n}$
- Output $\lambda_{*,1}$
- Output $\lambda_{*,n}$
**Block II**  
**High-Level Controller**

![Diagram](image_url)

- **Reception**
- **Q-Conn**
- **QUEUE**

- **HLC**

- **Equation:** $\Sigma(s_i+1) + s_{ts} + n_{out}$

- **Annotations:**
  - Push
  - $d_{in}$
input velo_1: Int64
input velo_2: Int64
output devi := abs(velo_1 - velo_2)
output lasting_devi := devi > 5
  ∧ devi.offset(by: -1, dft: 0) > 5
  ∧ devi.offset(by: -2, dft: 0) > 5
trigger lasting_devi "Lasting deviation in measured velocities."
output avg_devi @10mHz := devi.aggregate(over: 10min, using: avg)
trigger avg_devi > 4 "High average deviation."
input velo_1: Int64
input velo_2: Int64
output devi := abs(velo_1 - velo_2)
output lasting_devi := devi > 5
  ∧ devi.offset(by: -1, dft: 0) > 5
  ∧ devi.offset(by: -2, dft: 0) > 5
trigger lasting_devi "Lasting deviation in measured velocities."
output avg_devi @10mHz := devi.aggregate(over: 10min, using: avg)
trigger avg_devi > 4 "High average deviation."
input velo_1: Int64
input velo_2: Int64
output devi := abs(velo_1 - velo_2)
output lasting_devi := devi > 5
  ∧ devi.offset(by: -1, dft: 0) > 5
  ∧ devi.offset(by: -2, dft: 0) > 5
trigger lasting_devi "Lasting deviation in measured velocities."
output avg_devi @10mHz := devi.aggregate(over: 10min, using: avg)
trigger avg_devi > 4 "High average deviation."

{velo_2}: 0^{64+1}, velo_2, 1, ts, 00000
{velo_1, velo_2}: velo_1, 1, velo_2, 1, ts, 11100
\(t = 100s:\) 0^{64+1}, 0^{64+1}, ts, 00011
input velo_1: Int64
input velo_2: Int64
output devi := abs(velo_1 - velo_2)
output lasting_devi := devi > 5
  ∧ devi.offset(by: -1, dft: 0) > 5
  ∧ devi.offset(by: -2, dft: 0) > 5
trigger lasting_devi "Lasting deviation in measured velocities."
output avg_devi @10mHz := devi.aggregate(over: 10min, using: avg)
trigger avg_devi > 4 "High average deviation."
BLOCK II

PARALLEL COMPUTATION

idle

1

velo_1
velo_2

2.0

devi

2.1

lasting_devi
avg_devi

2.2

trigger 1
trigger 2

velo_1

avg_d

devi

velo_2

last_d

trigger 1

trigger 2
input velo_1: Int64
input velo_2: Int64
output devi ::= abs(velo_1 - velo_2)
output lasting_devi ::= devi > 5
  ∧ devi.offset(by: -1, dft: 0) > 5
  ∧ devi.offset(by: -2, dft: 0) > 5
trigger lasting_devi "Lasting deviation in measured velocities."
output avg_devi @10mHz ::= devi.aggregate(over: 10min, using: avg)
trigger avg_devi > 4 "High average deviation."
### Empirical Evaluation

<table>
<thead>
<tr>
<th></th>
<th>FF</th>
<th>LUT</th>
<th>MUX</th>
<th>CA</th>
<th>MULT</th>
<th>Pwr [W]</th>
<th>Time [μs]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Drone</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon</td>
<td>3036</td>
<td>3685</td>
<td>26</td>
<td>656</td>
<td>10</td>
<td>1.620</td>
<td>4.28</td>
</tr>
<tr>
<td>HLC</td>
<td>901</td>
<td>156</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>543</td>
<td>442</td>
<td>0</td>
<td>43</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLC</td>
<td>1281</td>
<td>2820</td>
<td>0</td>
<td>576</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon</td>
<td>1905</td>
<td>1533</td>
<td>23</td>
<td>226</td>
<td>23</td>
<td>1.570</td>
<td>3.20</td>
</tr>
<tr>
<td>HLC</td>
<td>550</td>
<td>161</td>
<td>0</td>
<td>37</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>330</td>
<td>342</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLC</td>
<td>895</td>
<td>927</td>
<td>0</td>
<td>161</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cmd Resp Par</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon</td>
<td>6379</td>
<td>13794</td>
<td>0</td>
<td>849</td>
<td>0</td>
<td>1.582</td>
<td>3.77</td>
</tr>
<tr>
<td>HLC</td>
<td>936</td>
<td>232</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>540</td>
<td>326</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLC</td>
<td>4903</td>
<td>13236</td>
<td>0</td>
<td>971</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cmd Resp Seq</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon</td>
<td>6909</td>
<td>14768</td>
<td>0</td>
<td>851</td>
<td>0</td>
<td>1.581</td>
<td>43.83</td>
</tr>
<tr>
<td>HLC</td>
<td>936</td>
<td>232</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>534</td>
<td>326</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLC</td>
<td>5433</td>
<td>14210</td>
<td>0</td>
<td>973</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Never injure humans.
2. Obey orders.
3. Protect yourself.
Never injure humans. Obey orders. Protect yourself.

Lola Specification

Compilation

Impl Monitor {
  while let Some(i) = get_input() {
    ... 
  }
}

High Level Code

observes

M O N I T O R

01010010
01010110
00110010
00110000
00110000
specs observes

Never injure humans. Obey orders. Protect yourself.

Lola Specification

Compilation

Impl Monitor {
  while let Some(i) = get_input() {
    ...
  }
}

Rust Code

MONITOR

01010010
01010110
00110010
00110000
00110000
never injure humans.

obey orders.

protect yourself.

Lola specification

Compilation

+ Annotation Generation

Impl Monitor {
    #[invariant = ... ]
    while let Some(i) = get_input() {
        ...
    }
}
```rust
impl Monitor {
    #[invariant = ... ]
    while let Some(i) = get_input() {
        ...
    }
}
```
**Block II**

**Software Compilation**

```
Impl Monitor {
    #[invariant = ... ]
    while let Some(i) = get_input() {
        ...
    }
}
```

**Viper**

- **Compilation**
- **Annotation Generation**
- **Lola Specification**
- **Rust Code**
- **Monitor**

- Verifies
- Observes

**Never injure humans.**
**Obey orders.**
**Protect yourself.**

**Lola Specification**

**Rust Code**

```
impl Monitor {
    #[invariant = ... ]
    while let Some(i) = get_input() {
        ...
    }
}
```
input alt

output tooHigh :=
   alt.offset(by: -1, dft: 0) > 500
   ∧ alt > 500
   ∧ alt.offset(by: +1, dft: 0) > 500

trigger tooHigh
input alt

output tooHigh :=
    alt.offset(by: -1, dft: 0) > 500
    ∧ alt > 500
    ∧ alt.offset(by: +1, dft: 0) > 500

trigger tooHigh
input alt

output tooHigh :=
  alt.offset(by: -1, dft: 0) > 500
  ∧ alt > 500
  ∧ alt.offset(by: +1, dft: 0) > 500

trigger tooHigh
input alt
output tooHigh :=
    alt.offset(by: -1, dft: 0) > 500 ∧ alt > 500 ∧ alt.offset(by: +1, dft: 0) > 500
trigger tooHigh
input alt
output tooHigh :=
  alt.offset(by: -1, dft: 0) > 500
  ∧ alt > 500
  ∧ alt.offset(by: +1, dft: 0) > 500
trigger tooHigh
**Block II**

**Shift & Memory Requirement**

**Def Shift:**

$$\Delta(s) = \max(0, \max\{w + \Delta(s') \mid (s,w,s') \in R\})$$

$$\Delta(alt) = 0$$

$$\Delta(\text{tooHigh}) = \Delta(\text{trig}) = 1$$

---

**Input:** alt

**Output:** tooHigh :=

- `alt.offset(by: -1, dft: 0) > 500`
- `alt > 500`
- `alt.offset(by: +1, dft: 0) > 500`

**Trigger:** tooHigh
**Def Shift:**

$$\Delta(s) = \max(0, \max\{w + \Delta(s') \mid (s,w,s') \in R\})$$

$$\Delta(\text{alt}) = 0$$

$$\Delta(\text{tooHigh}) = \Delta(\text{trig}) = 1$$

**Def Memory Requirement:**

$$\mu(s) = \max\{\Delta(s') - \Delta(s) - w \mid (s',w,s) \in E\}$$

$$\mu(\text{alt}) = 2$$

$$\mu(\text{tooHigh}) = \Delta(\text{trig}) = 0$$

---

**Block II**

**Shift & Memory Requirement**

**Input alt**

**Output tooHigh :=**

- alt.offset(by: -1, dft: 0) > 500
- alt > 500
- alt.offset(by: +1, dft: 0) > 500

**Trigger tooHigh**
**Block II**

**(In-)Fallible Accesses**

Sensors

\[ a_{1,1} \quad a_{1,2} \quad a_{1,3} \quad a_{1,4} \quad a_{1,5} \quad a_{1,6} \quad a_{1,7} \quad a_{1,8} \]

\[ tH_{2,1} \quad tH_{2,2} \quad tH_{2,3} \quad tH_{2,4} \quad tH_{2,5} \quad tH_{2,6} \quad tH_{2,7} \quad tH_{2,8} \]

**input alt**

**output tooHigh** :=

alt.offset(by: -1, dft: 0) > 500
∧ alt > 500
∧ alt.offset(by: +1, dft: 0) > 500

**trigger tooHigh**
(In-)Fallible Accesses

**sensors**

\[
\begin{array}{cccccccc}
  a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} & a_{1,7} & a_{1,8} \\
  tH_{2,1} & tH_{2,2} & tH_{2,3} & tH_{2,4} & tH_{2,5} & tH_{2,6} & tH_{2,7} & tH_{2,8}
\end{array}
\]

**input alt**

**output tooHigh :=**

\[
\text{alt.offset(by: -1, dft: 0) > 500} \\
\land \text{alt > 500} \\
\land \text{alt.offset(by: +1, dft: 0) > 500}
\]

**trigger tooHigh**
**BLOCK II  (IN-)FALLIBLE ACCESSES**

**Input alt**

**Output tooHigh :=**

- $\text{alt.offset(by: -1, dft: 0)} > 500$
- $\land \text{alt} > 500$
- $\land \text{alt.offset(by: +1, dft: 0)} > 500$

**Trigger tooHigh**
input `alt`

output `tooHigh` :=
- `alt.offset(by: -1, dft: 0) > 500`
- `alt > 500`
- `alt.offset(by: +1, dft: 0) > 500`

trigger `tooHigh`

fails @ t=1

fails @ t=|₁₀|
Block II (In-)Fallible Accesses

input alt

output tooHigh :=
\begin{align*}
&alt.\text{offset(by: -1, dft: 0)} > 500 \\
&\land alt > 500 \\
&\land alt.\text{offset(by: +1, dft: 0)} > 500
\end{align*}

fails @ t=1

fails @ t=|0|

trigger tooHigh
**Block II**  
**(In-)Fallible Accesses**

- **Input**: $\text{alt}$
- **Output**: $\text{tooHigh} := (\text{alt.offset(by: -1, dft: 0}) > 500 \land \text{alt} > 500 \land \text{alt.offset(by: +1, dft: 0}) > 500)$
- **Trigger**: $\text{tooHigh}$

**Fails @ t=1**
- $\text{alt.offset(by: -1, dft: 0}) > 500$
- $\text{alt} > 500$
- $\text{alt.offset(by: +1, dft: 0}) > 500$

**Fails @ t=|o|**
**Block II**

**(In-)Fallible Accesses**

Input `alt`

Output `tooHigh` :=

\[
alt.\text{offset(by: -1, dft: 0)} > 500 \land alt > 500 \land alt.\text{offset(by: +1, dft: 0)} > 500
\]

Never fails

Trigger `tooHigh`

Fails @ t=1

Fails @ t=0|
let alt_past = mem.get_alt(-1).unwrap_or(0);

let alt_future = mem.get_alt(+1).unwrap_or(0);

let alt_current = mem.get_alt_sync(0);

let tooHigh =
  alt_past > 500
  && alt_current > 500
  && alt_future > 500
let alt_past = mem.get_alt(-1).unwrap_or(0);

let alt_future = mem.get_alt(+1).unwrap_or(0);

let alt_current = mem.get_alt_sync(0);

let tooHigh =
    alt_past > 500 &&
    alt_current > 500 &&
    alt_future > 500
Three Phases

Prefix

Monitor Loop

Postfix
**Block II**  
**Three Phases**

### Prefix

```rust
def prefix() {
    a_{-1} > 500 \\
    \land a_0 > 500 \\
    \land a_{+1} > 500 
}
```

### Monitor Loop

```rust
while let Some(...) = get_input() {
    a_{-1} > 500 \\
    \land a_0 > 500 \\
    \land a_{+1} > 500 
}
```

### Postfix

```rust
def postfix() {
    a_{-1} > 500 \\
    \land a_0 > 500 \\
    \land a_{+1} > 500 
}
```
Block II  Three Phases

```r
fn prefix() {
    0 > 500 ∧ a₀ > 500 ∧ a₁ > 500
}

while let Some(...) = get_input() {
    a₋₁ > 500 ∧ a₀ > 500 ∧ a₁ > 500
}

fn postfix() {
    a₋₁ > 500 ∧ a₀ > 500 ∧ a₁ > 500
}
```
Block II

Three Phases

Fn prefix() {
    \( a_{-1} > 500 \)
    \( a_0 > 500 \)
    \( a_{+1} > 500 \)
}

Fn postfix() {
    \( a_{-1} > 500 \)
    \( a_0 > 500 \)
    \( a_{+1} > 500 \)
}

While let Some(...) = get_input() {
    \( a_{-1} > 500 \)
    \( a_0 > 500 \)
    \( a_{+1} > 500 \)
}
**Block II**

**Three Phases**

---

**Prefix**

\[
\text{fn prefix()} \{ \\
0 > 500 \\
\land a_0 > 500 \\
\land a_1 > 500 \\
\}
\]

---

**Monitor Loop**

\[
\text{while let Some(...)} \quad \text{=} \quad \text{get_input()} \{ \\
\qquad a_{-1} > 500 \\
\qquad \land a_0 > 500 \\
\qquad \land a_{+1} > 500 \\
\}
\]

---

**Postfix**

\[
\text{fn postfix()} \{ \\
\quad a_{+1} > 500 \\
\quad \land a_0 > 500 \\
\quad \land 0 > 500 \\
\}
\]
I. **Eradicate Most Conditionals**

II. **Replace Memory Accesses with Constants**

**Prefix**

```
fn prefix() {
  0 > 500 ∧ a₀ > 500 ∧ a₁ > 500
}
```

**Monitor Loop**

```
while let Some(...) = get_input() {
  a₋₁ > 500 ∧ a₀ > 500 ∧ a₊₁ > 500
}
```

**Postfix**

```
fn postfix() {
  a₋₁ > 500 ∧ a₀ > 500 ∧ 0 > 500
}
```
Block II  Performance Benefit

Interpreter  438ns  1.535μs (1.4%)
Compilation  6ns  63ns (4%)
**Block II** Verification — Idea

**LOLA**

\[ t = 0 \]

\[ \text{stream} \triangleq \text{infinite sequence of values} \]

**RUST**
Block II Verification — Idea

**LOLA**

\[ t = 0 \]

\[ \text{stream} \triangleq \text{infinite sequence of values} \]

**RUST**

\[ \mu(s) \]

\[ \text{memory} \triangleq \text{finite excerpt of stream} \]
**Block II Verification — Idea**

- **Stream**: $\triangleq$ infinite sequence of values

**Lola**

- $t = 0$

**Viper**

- $t = 0$

**Rust**

- $\mu(s)$

**Memory**: $\triangleq$ finite excerpt of stream

**Ghost Memory**
**Block II**

**Verification — Idea**

**Stream** = infinite sequence of values

**Lola**

$t = 0$

**Viper**

$t = 0$

**Rust**

$\mu(s)$

**Ghost Memory**

**Memory** = finite excerpt of stream
**Block II Verification — Idea**

- **Stream**: $\triangleq$ infinite sequence of values
- **LOLA**: 
  - $t = 0$
  - Memory: $\mu(s) = 1$
- **Viper**: 
  - $t = 0$
  - Memory: $\mu(s) = 2$
- **Rust**: 
  - Memory: $\triangleq$ finite excerpt of stream

*Diagram showing the relationship between LOLA, Viper, and Rust.*
while let Some(input) = get_input() {
    mem.add_input(&input);
    [[ EVALUATION LOGIC ]]
    mem.store(new_tooHigh);
    gm.store(new_tooHigh);
    if trigger_1 { emit( trigger_1_msg) }
}
while let Some(input) = get_input() {
    mem.add_input(&input);

    [[ EVALUATION LOGIC ]]
    mem.store(new_tooHigh);
    gm.store(new_tooHigh);

    if trigger_1 { emit( trigger_1_msg) }
}
```
#![invariant="forall i: usize ::
    (0 <= i && i < μ(a))
    ==> mem.get_a(i) == gm.get_a(iter - i)"
]

#![invariant="new_tooHigh == gm.get_a(iter - 2) > 500 ∧ ..."]

while let Some(input) = get_input() {
    mem.add_input(&input);

    [[ EVALUATION LOGIC ]]

    mem.store(new_tooHigh);
    gm.store(new_tooHigh);

    if trigger_1 { emit( trigger_1_msg) }
}
```
Block II: Viability
Block II  Viability
Block II Viability

altitutde

![Graph comparing Time [s] vs Memory [MB]](image1)

![Graph comparing Time [s] vs Memory [MB]](image2)
Detected implicit assumption on input stream!
input time
input sensor

output δtime :=
    abs(time - time.offset(by: -1, dft: 0))
output δsensor :=
    abs(sensor - sensor.offset(by: -1, dft: 0))
output diff := δsensor / δtime

Detected implicit assumption on input stream!
input time
input sensor

output $\delta$time :=
  abs(time - time.offset(by: -1, dft: 0))
output $\delta$sensor :=
  abs(sensor - sensor.offset(by: -1, dft: 0))
output diff := $\frac{\delta$sensor}{\delta$time}$

Detected implicit assumption on input stream!
Detected implicit assumption on input stream!
Detected implicit assumption on input stream!

On corrected spec:

- 6 – 16min
- 1.38 – 1.66GB
- 2 T/O (10%)
- 4 fails (20%)
I. SUCCESSFULLY DETECTED SPECIFICATION ERROR
II. VERIFIED MONITORS FOR COMPLEX SPECIFICATIONS

Detected implicit assumption on input stream!

On corrected spec:

6 – 16min
1.38 – 1.66GB
input alt
input pressure

output tooHigh :=
  alt.offset(by: -1, dft: 0) > 500
  ∧ alt > 500
  ∧ alt.offset(by: +1, dft: 0) > 500

trigger tooHigh
trigger pressure < 0
let (v_1, ..., v_n) = crossbeam::scope(|scope| {
    let handle_tooHigh = scope.spawn(move |_| {
        eval_tooHigh(&memory)
    });
    let handle_trigger_2 = scope.spawn(move |_| {
        eval_trigger_2(&memory)
    });
    (
        handle_s1.join().unwrap(),
        ..., 
        handle_sn.join().unwrap()
    )
}).unwrap()
I. Enabling Software Parallelization is a Double-Edged Sword.
II. Hardware Parallelization is a No-Brainer.
Block II

Summary SW Compilation

Viper

Never injure humans. Obey orders. Protect yourself.

Compilation + Annotation Generation

Lola Specification

Impl Monitor {
    #[invariant = ... ]
    while let Some(i) = get_input() {
        ...
    }
}

Rust Code

Monitors observes

Verifies

 specifications

M

ONITOR

01010010 01010110 00110010 00110000
Block III Integration

Block I
- Type-Check
- Interpretation
- Specification

Block II
- Hardware
- Software

Block III
- Post-Mortem
- Integration
Block III: DLR’s superARTIS
Block III: DLR’s superARTIS
Block III DLR’s SUPERARTIS

Camera → Lidar → IMU

GNSS → ... → Position Estimation

→ Logger

→ HD
BLOCK III  DLR’S SUPERARTIS

- Camera
- GNSS
- Lidar
- IMU
- Position Estimation
- Logger
- Monitor
- HD
Snooping v Messages

Factors:
- Bus Utilization / Bus Allocation
- Resource Availability
BLOCK III  SPECS I: (CROSS-) VALIDATION

- Camera
- GNSS
- Lidar
- IMU
- Position Estimation
- Logger
- HD
Block III: Specs II: Geo-Fencing
**Block III  Specs II: Geo-Fencing**

I. **Arithmetically Challenging**

II. **Highly Parallel**
## Block III  Resource Consumption

<table>
<thead>
<tr>
<th>FF (%)</th>
<th>LUT</th>
<th>MUX</th>
<th>Pwr Idle [mW]</th>
<th>Pwr Peak [W]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,853</td>
<td>26,181</td>
<td>4</td>
<td>149</td>
<td>1.871</td>
</tr>
<tr>
<td>4,792</td>
<td>34,630</td>
<td>104</td>
<td>156</td>
<td>2.085</td>
</tr>
<tr>
<td>3,441</td>
<td>23,261</td>
<td>99</td>
<td>150</td>
<td>1.911</td>
</tr>
</tbody>
</table>
### Block III Resource Consumption

<table>
<thead>
<tr>
<th>FF (%)</th>
<th>LUT</th>
<th>MUX</th>
<th>Pwr Idle [mW]</th>
<th>Pwr Peak [W]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,853 (3)</td>
<td>26,181 (71)</td>
<td>4</td>
<td>149</td>
<td>1.871</td>
</tr>
<tr>
<td>4,792 (5)</td>
<td>34,630 (67)</td>
<td>104</td>
<td>156</td>
<td>2.085</td>
</tr>
<tr>
<td>3,441 (4)</td>
<td>23,261 (46)</td>
<td>99</td>
<td>150</td>
<td>1.911</td>
</tr>
<tr>
<td>FF (%)</td>
<td>LUT</td>
<td>MUX</td>
<td>Pwr Idle [mW]</td>
<td>Pwr Peak [W]</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-----</td>
<td>---------------</td>
<td>--------------</td>
</tr>
<tr>
<td>2,853</td>
<td>26,181 (71)</td>
<td>4</td>
<td>149</td>
<td>1.871</td>
</tr>
<tr>
<td>4,792</td>
<td>34,630 (67)</td>
<td>104</td>
<td>156</td>
<td>2.085</td>
</tr>
<tr>
<td>3,441</td>
<td>23,261 (46)</td>
<td>99</td>
<td>150</td>
<td>1.911</td>
</tr>
</tbody>
</table>
## Block III Resource Consumption

<table>
<thead>
<tr>
<th>FF (%)</th>
<th>LUT</th>
<th>MUX</th>
<th>Pwr Idle [mW]</th>
<th>Pwr Peak [W]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,853</td>
<td>26,181</td>
<td>4</td>
<td>149</td>
<td>1.871</td>
</tr>
<tr>
<td>4,792</td>
<td>34,630</td>
<td>104</td>
<td>156</td>
<td>2.085</td>
</tr>
<tr>
<td>3,441</td>
<td>23,261</td>
<td>99</td>
<td>150</td>
<td>1.911</td>
</tr>
</tbody>
</table>
### Block III Resource Consumption

<table>
<thead>
<tr>
<th>FF (%)</th>
<th>LUT</th>
<th>MUX</th>
<th>Pwr Idle [mW]</th>
<th>Pwr Peak [W]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,853</td>
<td>26,181 (71)</td>
<td>4</td>
<td>149</td>
<td>1.871</td>
</tr>
<tr>
<td>4,792</td>
<td>34,630 (67)</td>
<td>104</td>
<td>156</td>
<td>2.085</td>
</tr>
<tr>
<td>3,441</td>
<td>23,261 (46)</td>
<td>99</td>
<td>150</td>
<td>1.911</td>
</tr>
</tbody>
</table>
I. (Quantitative) Stream-based RV superior to Boolean Verdicts

II. Monitor naturally refines and filters data.

III. Access to crucial data.
**Block III  Conclusion**

**Block I**
- Type-Check
- Interpretation
- Specification

**Block II**
- Hardware
- Software

**Block III**
- Post-Mortem
- Integration

---

56
For a Monitor To Show its Full Potential, It Needs To Be Co-Developed With the CPS!