Exposing Design Flaws in Shared-Clock Systems using TLA+

Russell Mull, Auxon Corporation

TLA+ Conf

September 12, 2019

About Me

Russell Mull

Software Engineer, Auxon Corporation

How safe electronic systems are designed

How safe electronic systems are designed

- Decide what matters (safety requirements)

How safe electronic systems are designed

- Decide what matters (safety requirements)
- Decide how much it matters (Assign a Safety Integrity Level - SIL)

How safe electronic systems are designed

- Decide what matters (safety requirements)
- Decide how much it matters (Assign a Safety Integrity Level - SIL)
- Analyze the parts of the system that matter (Fault Tree Analysis)

How safe electronic systems are designed

- Decide what matters (safety requirements)
- Decide how much it matters (Assign a Safety Integrity Level - SIL)
- Analyze the parts of the system that matter (Fault Tree Analysis)
- Not good enough? Add redundancy.

Example: Industrial Press

Example: Industrial Press

- Safety requirement: Turn off press with emergency stop button

Example: Industrial Press

- Safety requirement: Turn off press with emergency stop button
- SIL: 4

Example: Industrial Press

- Safety requirement: Turn off press with emergency stop button
- SIL: 4
- Fault tree: the actuator is only SIL 3

Example: Industrial Press

- Safety requirement: Turn off press with emergency stop button
- SIL: 4
- Fault tree: the actuator is only SIL 3
- Redundancies: use two, design a SIL 4 failover mechanism

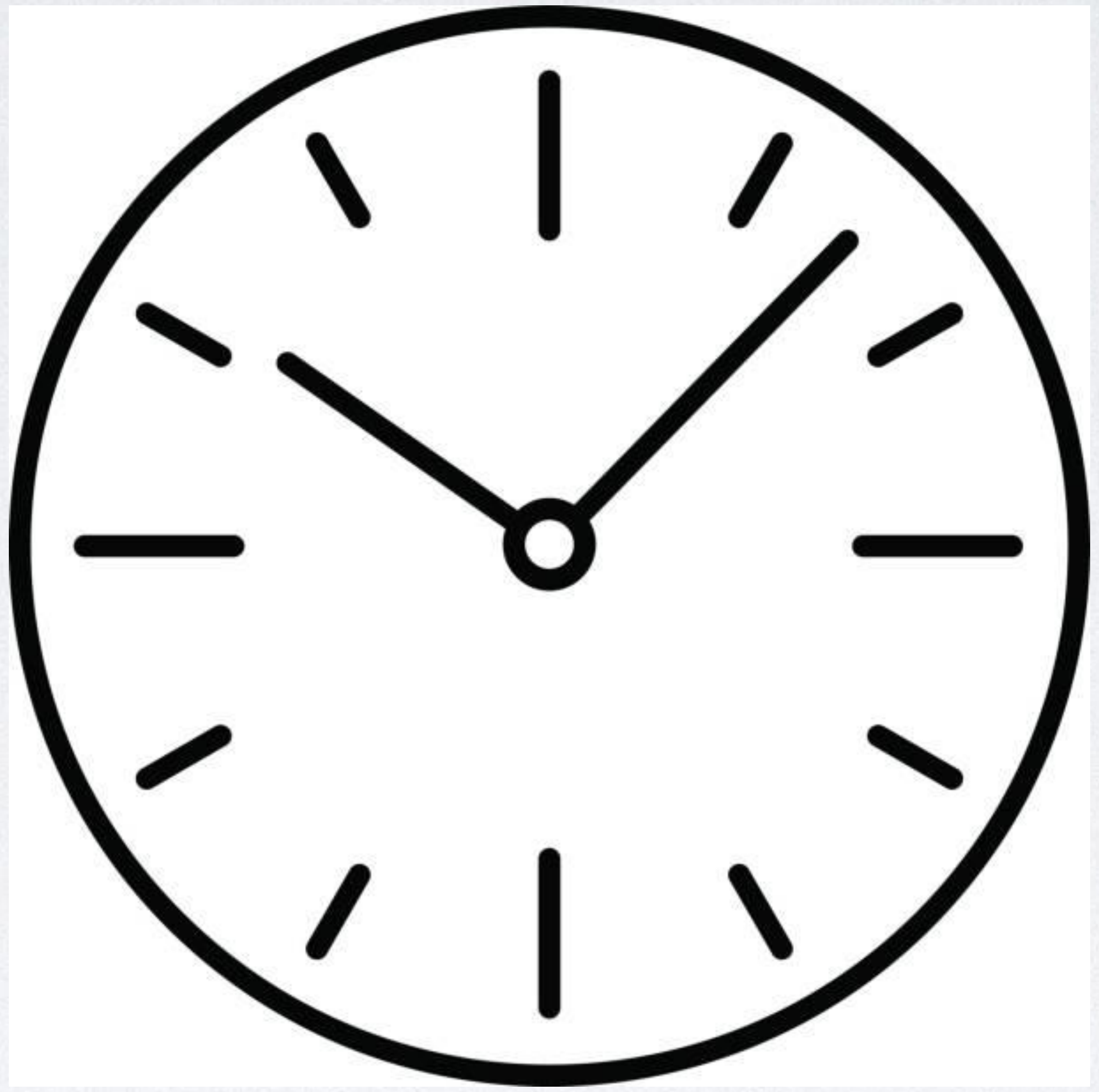
Functional Safety

- IEC 61508
- Power plants, chemical plants, cars, trains, heavy machinery, etc.

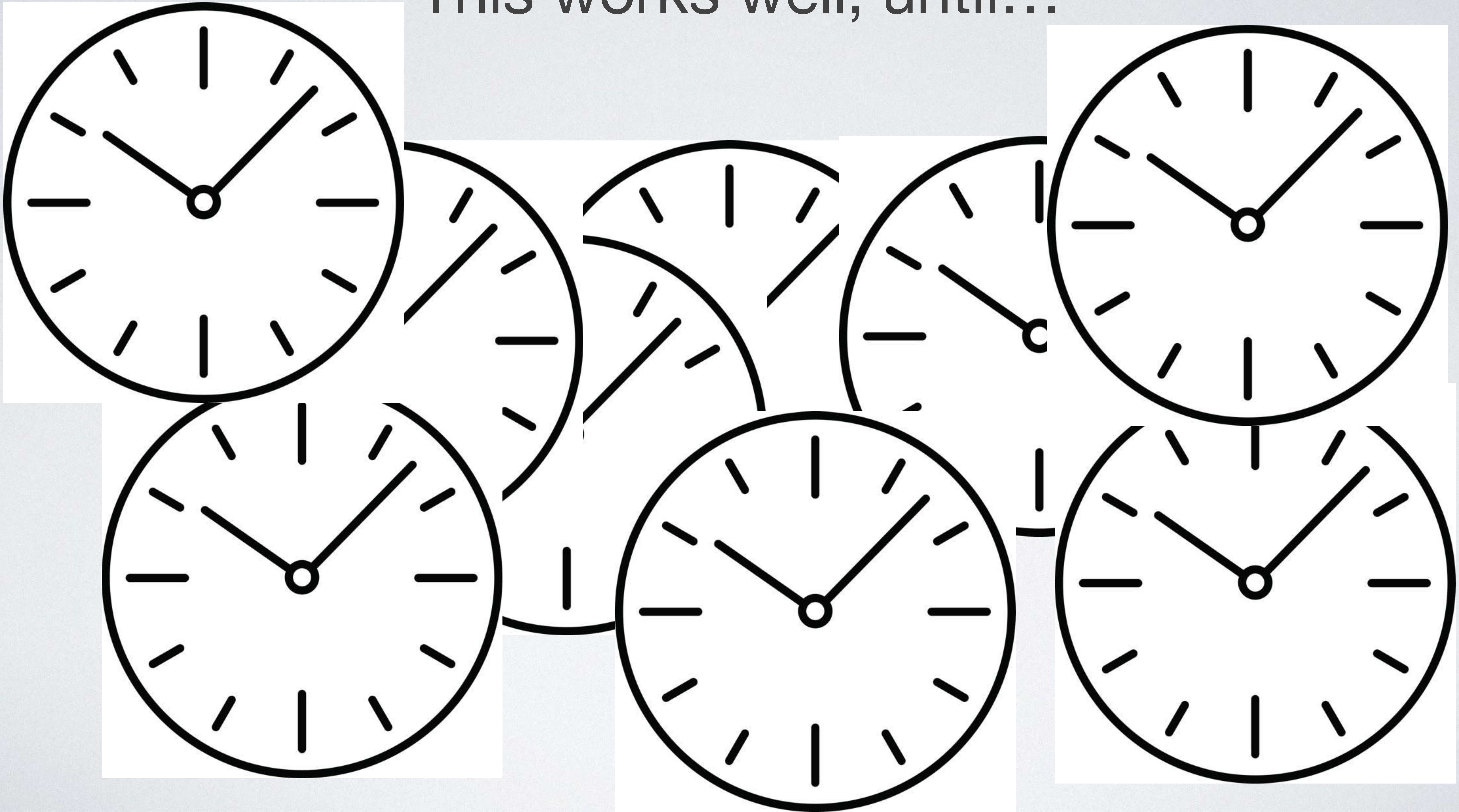


This works well, until…

This works well, until…



This works well, until…



This works well, until…

In software, shared clock failures are lumpy and unpredictable

The story of a system made from lots of computers, sensors, actuators, and clocks

A client project for [redacted]

A client project for [redacted]

- Can’t say anything specific

A client project for [redacted]

- Can’t say anything specific
- Relies fundamentally on a common timebase

A client project for [redacted]

- Can’t say anything specific
- Relies fundamentally on a common timebase
- Appeared to be vulnerable to drift

My Goal: Demonstrate the problem

A naïve model

A naïve model

```
VARIABLES node_clock, system
Nodes == { "A", "B", "C" }
```

A naïve model

```
VARIABLES node_clock, system
Nodes == { "A", "B", "C" }

Init ==
  /\ node_clock = [ n \in Nodes |-> 0 ]
  /\ system = ...
```

A naïve model

```
VARIABLES node_clock, system
Nodes == { "A", "B", "C" }

Init ==
  /\ node_clock = [ n \in Nodes |-> 0 ]
  /\ system = ...

Next ==
  \/ \E node \in DOMAIN node_clock:
       /\ node_clock' = [node_clock EXCEPT ![node] = @ + 1]
       /\ UNCHANGED << system >>
```

A naïve model

```
VARIABLES node_clock, system
Nodes == { "A", "B", "C" }

Init ==
  /\ node_clock = [ n \in Nodes |-> 0 ]
  /\ system = ...

Next ==
  \/ \E node \in DOMAIN node_clock:
     /\ node_clock' = [node_clock EXCEPT ![node] = @ + 1]
     /\ UNCHANGED << system >>
  \/ /\ SystemStep(system)
     /\ UNCHANGED << node_clock >>
```

A naïve model

```
VARIABLES node_clock, system
Nodes == { "A", "B", "C" }

Init ==
  /\ node_clock = [ n \in Nodes |-> 0 ]
  /\ system = ...

Next ==
  \/ \E node \in DOMAIN node_clock:
     /\ node_clock' = [node_clock EXCEPT ![node] = @ + 1]
     /\ UNCHANGED << system >>
  \/ /\ SystemStep(system)
     /\ UNCHANGED << node_clock >>
  \/ SyncClocks(node_clock, system)
```

A naïve model

```
VARIABLES node_clock, system
Nodes == { "A", "B", "C" }

Init ==
  /\ node_clock = [ n \in Nodes |-> 0 ]
  /\ system = ...

Next ==
  \/ \E node \in DOMAIN node_clock:
       /\ node_clock' = [node_clock EXCEPT ![node] = @ + 1]
       /\ UNCHANGED << system >>
  \/ /\ SystemStep(system)
     /\ UNCHANGED << node_clock >>
  \/ SyncClocks(node_clock, system)

SystemStep(s) == ...
SyncClocks(cs,s) == ...
```

This approach is not great.

This approach is not great.

- Massive state explosion

This approach is not great.

- Massive state explosion
- Customer doesn’t care about the sync protocol

Model the drift, not the sync

Drift Modeling (1)

```
CONSTANTS SIMULATED_CYCLES, BOUNDED_DRIFT
VARIABLES node_clock, system, global_clock
```

Drift Modeling (1)

```
CONSTANTS SIMULATED_CYCLES, BOUNDED_DRIFT
VARIABLES global_clock, node_clock, system

Nodes == { "A", "B", "C" }
```

Drift Modeling (1)

```
CONSTANTS SIMULATED_CYCLES, BOUNDED_DRIFT
VARIABLES global_clock, node_clock, system

Nodes == { "A", "B", "C" }

Init ==
  /\ global_clock = 0
  /\ node_clock = [ n \in Nodes |-> 0 ]
```

Drift Modeling (1)

```
CONSTANTS SIMULATED_CYCLES, BOUNDED_DRIFT
VARIABLES global_clock, node_clock, system

Nodes == { "A", "B", "C" }

Init ==
  /\ global_clock = 0
  /\ node_clock = [ n \in Nodes |-> 0 ]

Next ==
  \/ ClockStep /\ UNCHANGED system
  \/ SystemStep /\ UNCHANGED global_clock /\ UNCHANGED node_clock
```

Drift Modeling (1)

```
CONSTANTS SIMULATED_CYCLES, BOUNDED_DRIFT
VARIABLES global_clock, node_clock, system

Nodes == { "A", "B", "C" }

Init ==
  /\ global_clock = 0
  /\ node_clock = [ n \in Nodes |-> 0 ]

Next ==
  \/ ClockStep /\ UNCHANGED system
  \/ SystemStep /\ UNCHANGED global_clock /\ UNCHANGED node_clock

SystemStep == ...
```

Drift Modeling (2)

```
ClockStep ==
```

Drift Modeling (2)

```
ClockStep ==
  \* Tick the global clock
  \/ /\ global_clock' = global_clock + 1
     /\ UNCHANGED << node_clock >>
     /\ ClockDriftInBounds(global_clock', node_clock)
```

Drift Modeling (2)

```
ClockStep ==
  \* Tick the global clock
  \/ /\ global_clock' = global_clock + 1
     /\ UNCHANGED << node_clock >>
     /\ ClockDriftInBounds(global_clock', node_clock)

  \* Tick a node clock
  \/ \E node \in DOMAIN node_clock:
     /\ node_clock' = [node_clock EXCEPT ![node] = @ + 1]
     /\ UNCHANGED << global_clock >>
     /\ ClockDriftInBounds(global_clock, node_clock')
```

Drift Modeling (3)

```
ClockDriftInBounds(g, n) ==
  /\ g <= SIMULATED_CYCLES
  /\ \A node \in DOMAIN n :
     /\ n[node] <= SIMULATED_CYCLES
     /\ Abs(c[node] - g) <= BOUNDED_DRIFT
```

This works better

This works better

- Narrower state space

This works better

- Narrower state space
- Directly addresses relevant failure domain

The system was more vulnerable to drift than previously thought

Delivering a Model

- Literate PDF
- Makefile / .cfg file
- Config Instructions

TLA+ is tricky to use this way

- Difficult setup
- Easier development
- Easier delivery

Give models to your customers

Extending the technique

- Asymmetric Drift
- Action on Tick
- Cyclical Clock

Closing Thoughts

Closing Thoughts

- Fake a real clock

Closing Thoughts

- Fake a real clock
- Bound the drift

Closing Thoughts

- Fake a real clock
- Bound the drift
- Give models to your customers

Closing Thoughts

- Fake a real clock
- Bound the drift
- Give models to your customers
- I owe Hillel Wayne a great debt

Russell Mull

@mullr

russell@auxon.io