Logic Model Checking of Unintended Acceleration Claims in the 2005 Toyota Camry Electronic Throttle Control System

Ed Gamble & Gerard Holzmann

Jet Propulsion Laboratory

California Institute of Technology

19 October 2011

Introduction

Toyota had the death of a CHP Officer to explain

Toyota had a growing number of unsubstantiated reports of ‘Sudden Unintended Acceleration’

The US Department of Transportation promised to ‘get into the weeds’

Engine Control Systems share similarities with Spacecraft Control Systems

- Safety/Mission Critical
- Asynchronous
- Hard Real-Time
- Fault Tolerant
- Resource Constrained
- Environmentally Challenged

The JPL 'Laboratory for Reliable Software' Knows How to Find Subtle Software and System Problems

Core Technologies

Static Analysis (software)

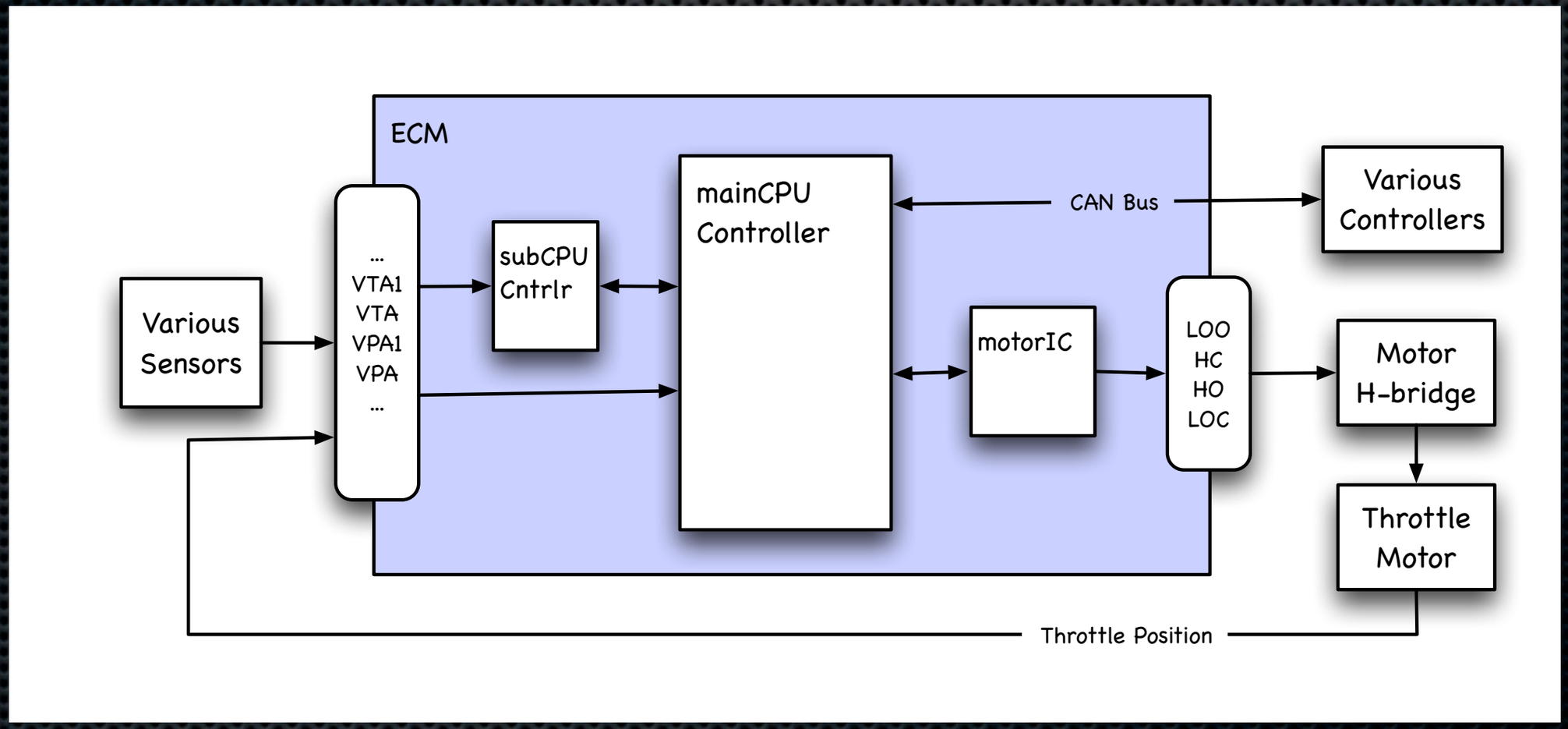
Logic Model Checking (software and systems)

The US DOT, through NASA, (Michael Aguilar, NESC) established a team

- Software analysis included JPL and Ames
- Other NASA teams: …, radiation, human factors
- Worked at Toyota HQ
- Developed IP protection protocols
- Toyota provided domain experts (+ Japanese trans.)
- Got the ’05 Camy L4 Software quickly!

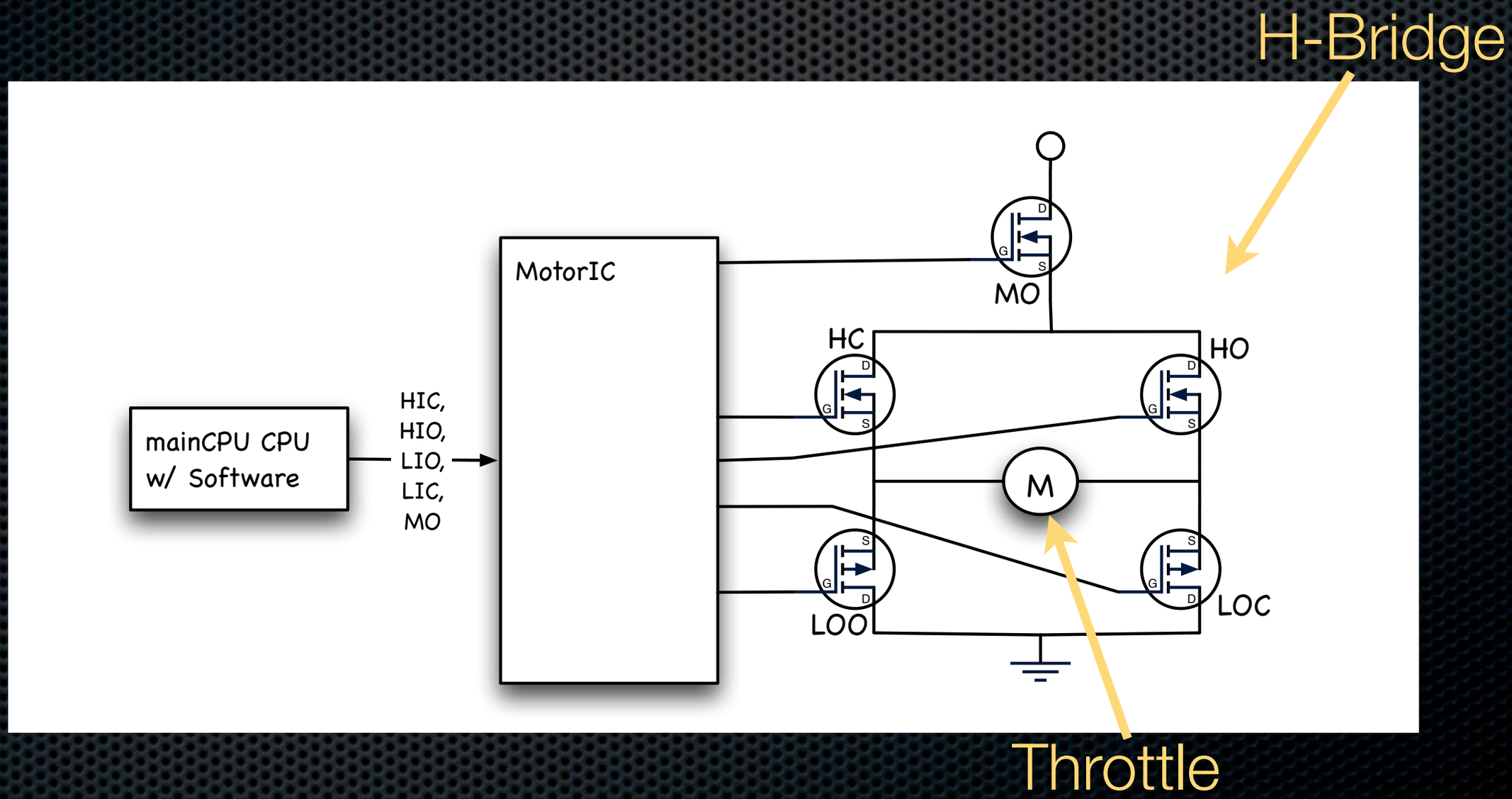
Throttle Control System And Software

Throttle Control System Overview

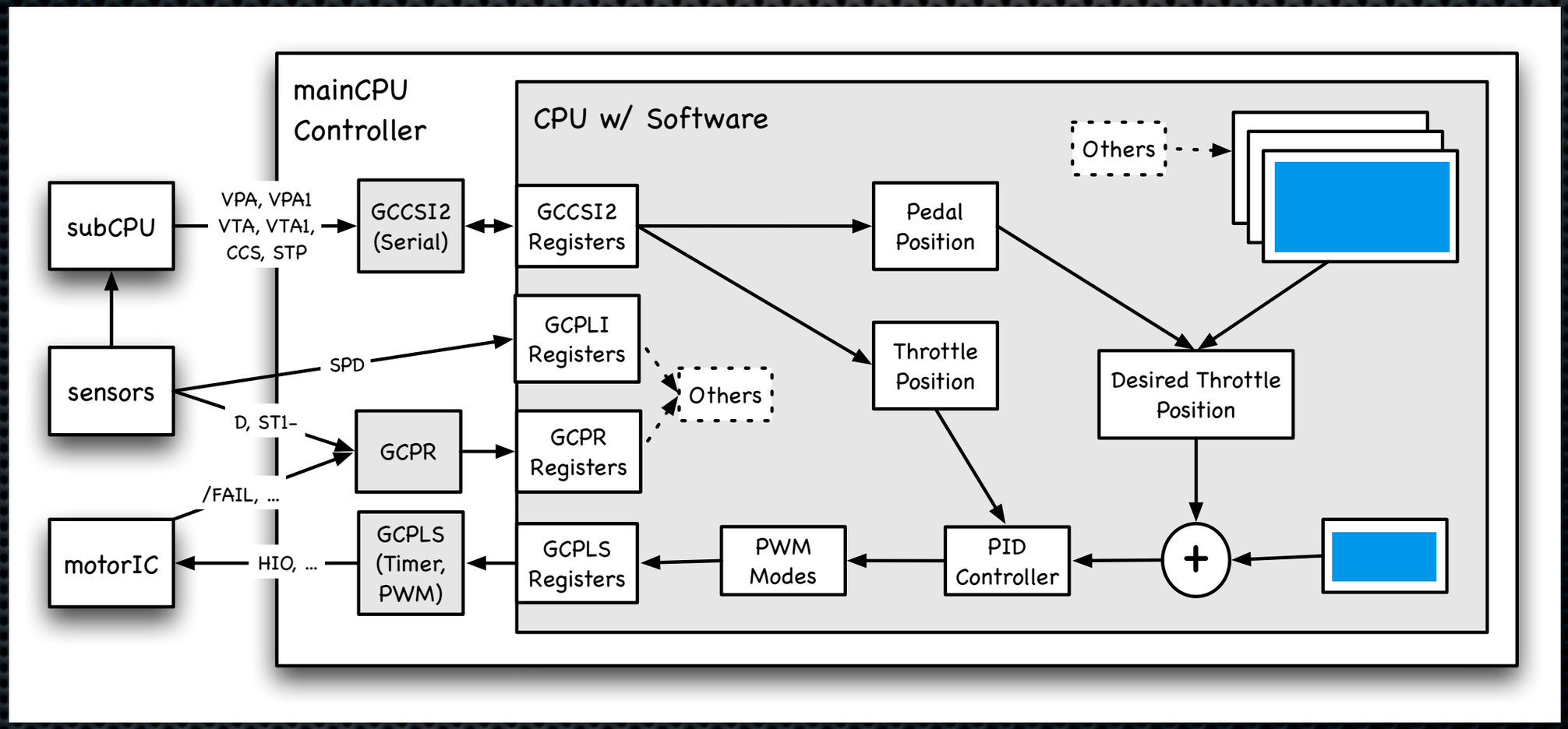


[Figure 6.4.1-1 - Abstracted]

Throttle Control Motor Drive IC



Throttle Control Software Overview



[Appendix A Figures - Merged]

Throttle Control Software/ System Properties

- NEC V850 E1 (32 βH) embedded controller
- ANSI C, GreenHills tools
  - files, kSLOC
- OSEK OS (auto. std.)
  - tasks, priorities, shared resource
- Two rate classes
  - Time: Hz / ms
  - Crank: rpm
- 'Product Line' code
  - strict inheritance, coding rules and process rqmts; in Japanese

Unintended Acceleration Properties and Implications

- Properties
  - Very Low Probability
  - Unreproducible
  - No persistent damage
    - No OBD II codes
- Implications (Software)
  - Masquerade as correct
  - Later in the processing
  - Multi-factor boundary condition

Static Analysis

Why Static Analysis?

- Static Analysis can:
  - Identify **legitimate** software errors
  - Be performed relatively **quickly** and **uniformly**
  - Be **customized** for the software coding style
  - Provide a general **sense*** of the software quality

Static Analysis Tools

- A suite of Static Analysis tools were applied:
  - CodeSonar: Augmented with ‘JPL Coding Standards’
  - Coverity: Augmented with ‘Power of Ten’ rules
  - GCC: Strict compilation flags
  - Uno: As provided

Static Analysis Results

- Can’t tell you about these...

Static Analysis Results

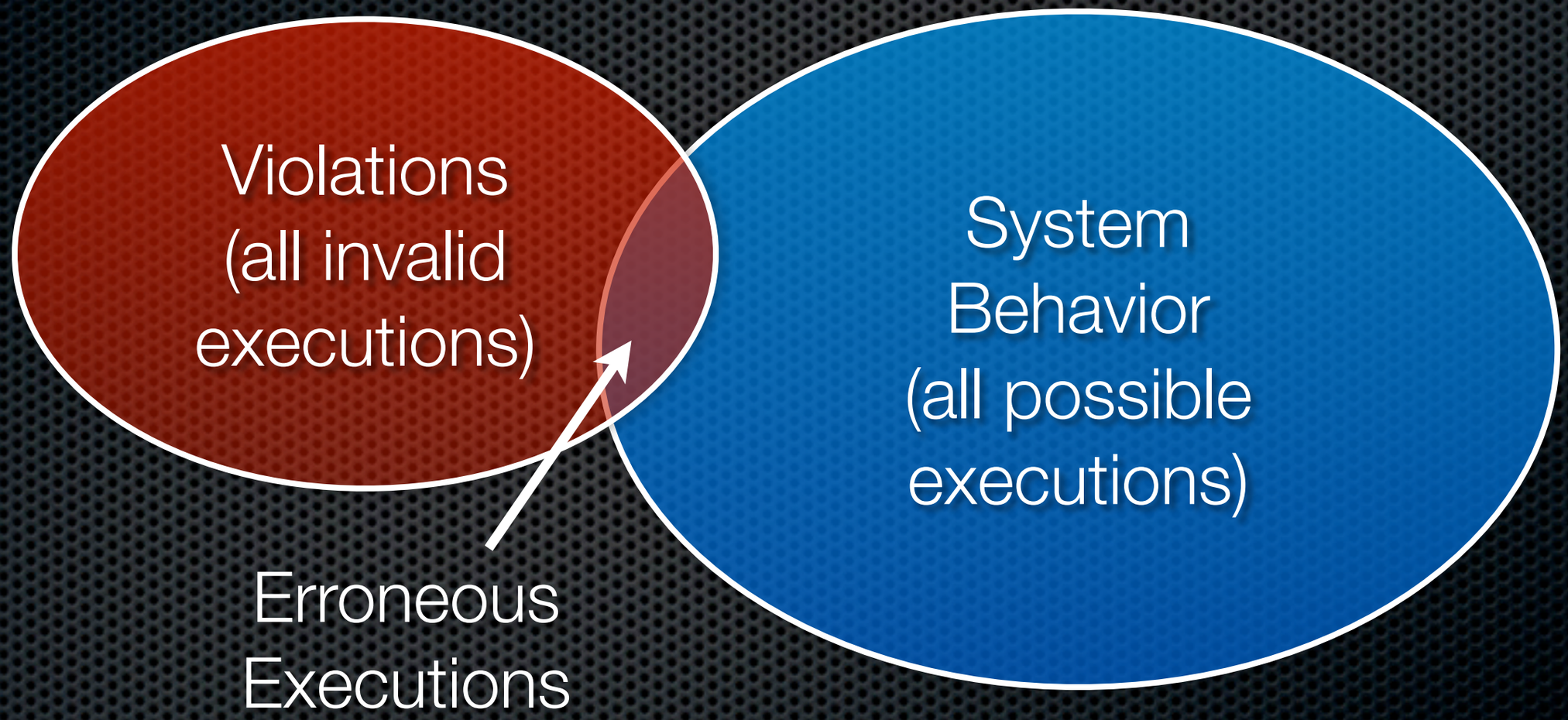
- The Static Analysis results are redacted in the public NASA report to the DOT. Possible reasons:
  - A comparison between EETCS software and spacecraft FSW was considered inappropriate,
  - An assessment of software risk, based on the statistics of static analysis results, was considered inappropriate given that a cause for SUA was not identified.

Logic Model Checking

What is Logic Model Checking?

- Verify the correctness of a system using a model of the system's behavior.
  - **Logic**: Define system correctness using linear temporal logic (LTL)
  - **Model**: Develop a model of the system behavior using PROMELA (Process Meta Language)
  - **Checking**: Exhaustively search the model behavior for violations of the correctness claims
    - Report behavior 'trails'

What is Logic Model Checking?



explore all executions; definitive conclusions; no probability

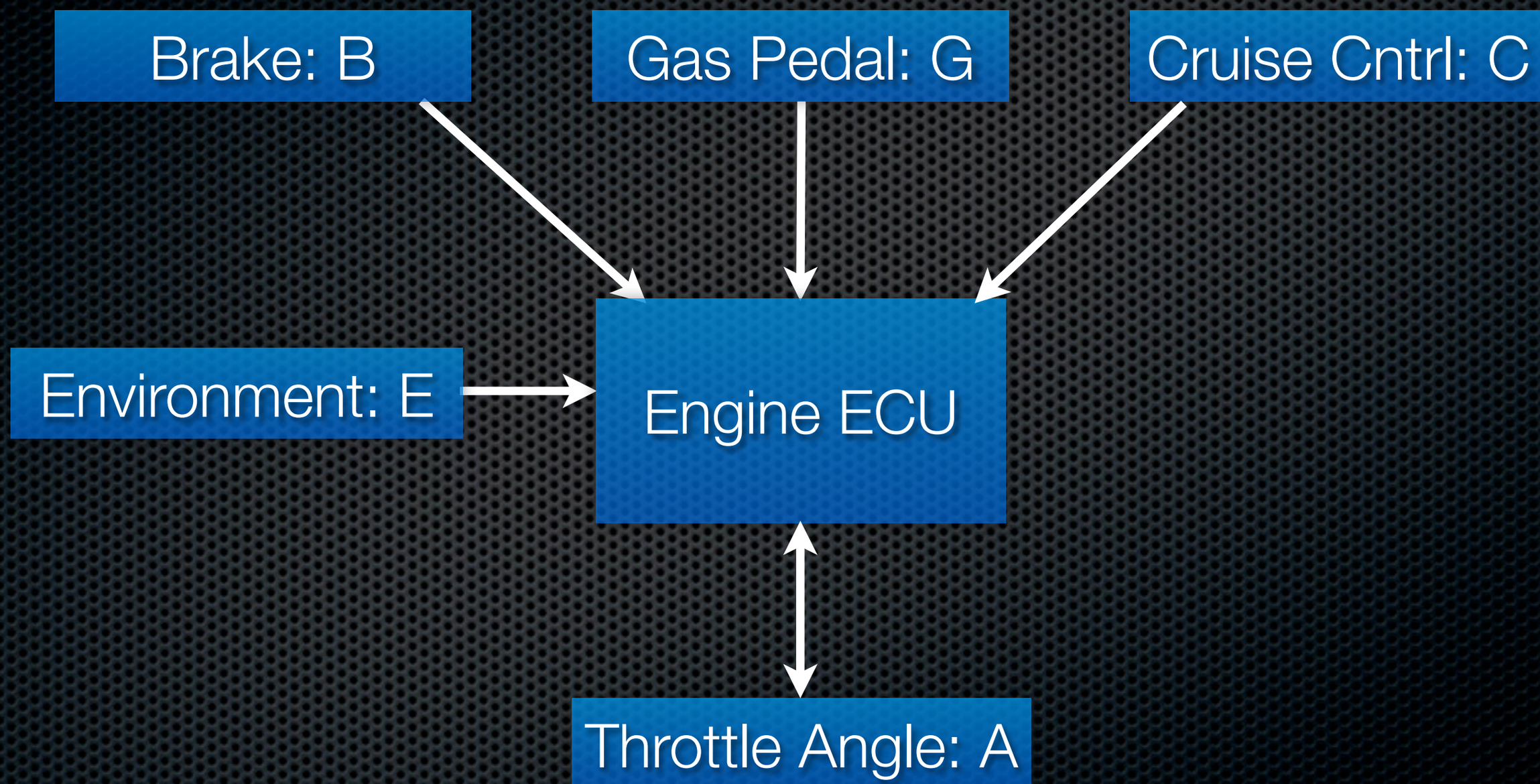
Verifications w/ **SPIN** and **SWARM** technology

Why Logic Model Checking?

- Logic Model Checking (with SPIN verification) can:
  - Explore **all** possible **states** of the model
  - Provide definitive **right** or **wrong** conclusions
  - Yield conclusions **without** regard to probability
  - Express statements of correct behavior easily

Logic Model Analyses

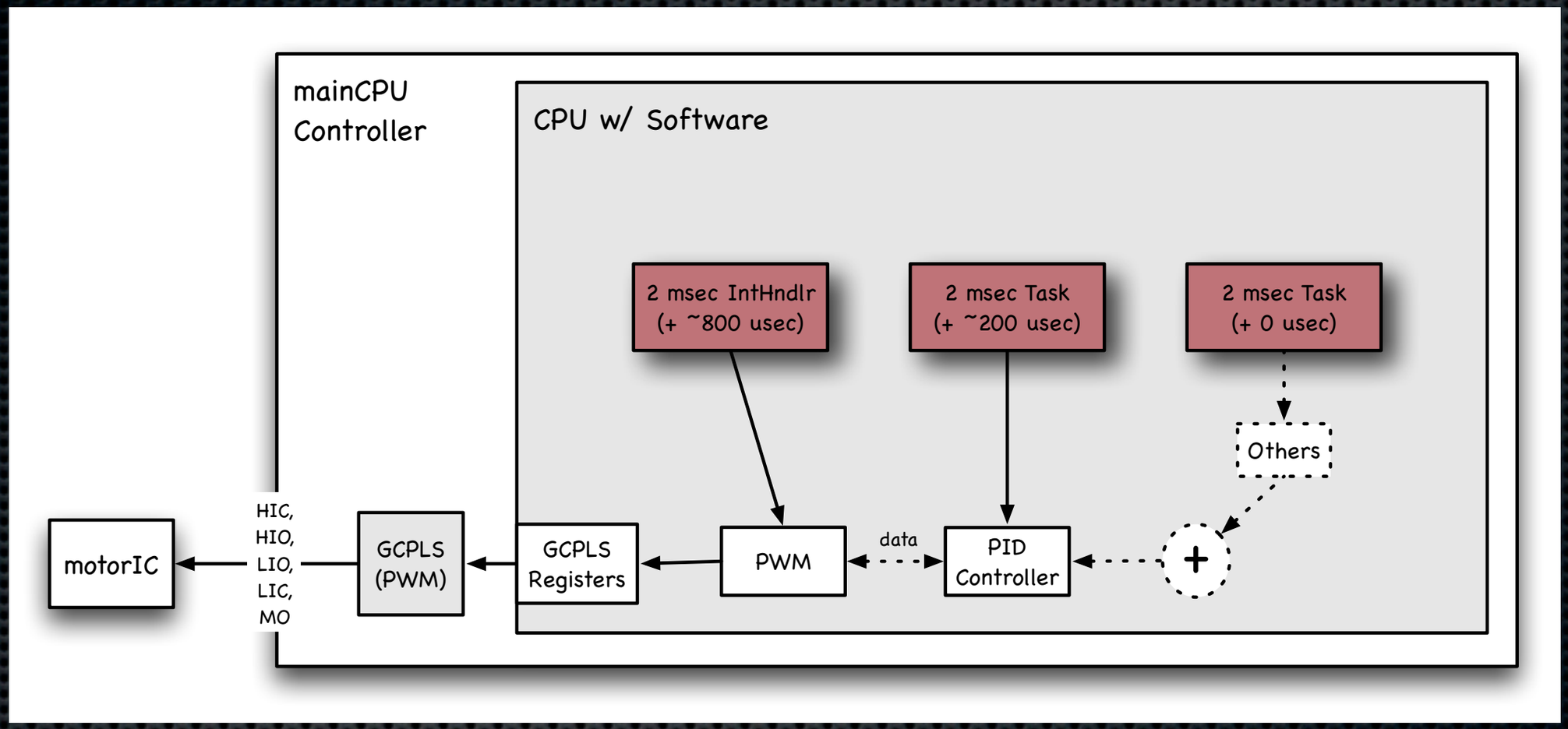
System Context



Logic Models Implemented

Logic Model	Type	Conclusion
Interrupt Enable / Disable Pairing	Computation	Verified
Accel. Pedal Learning	Computation	Inconclusive
Sensor Input	I/O	Potential Issue
Motor Drive IC	Computation	Verified
Port Reg. Input	I/O	Verified
PWM Functionality	Computation	Potential Issue

PWM Driver



The PWM drives the motor, through the MotorIC, open or closed based on a desired throttle angle

PWM Correctness

1. PWM output signals never eventually lead to an H-bridge short.
2. PWM output signals never eventually always drive the throttle wide open unless demanded.

PWM Claim Implementations

```
/*
 * Verify that the H-bridge short paths never occur.
 */
#define p ((MHP_IS_ON && MLP_IS_ON) || (MHM_IS_ON && MLM_IS_ON))

never {    /* <>p */
T0_init:
        if
        :: ((p)) -> goto accept_all
        :: (1) -> goto T0_init
        fi;
accept_all:
        skip
}
```

```
/*
 * Verify that the throttle plate is never wide open continuously
 * unless the inputs (duty cycle is +100%) demand it.
 */
#define p (MHM_IS_ON && MLP_IS_ON)
#define q (s2g_gamtr_gaduty == DUTY_100 && big_gamtr_gaxdtymn == OFF)

never {
T0_init:
        if
        :: (! ((q)) && (p)) -> goto accept_S4
        :: (1) -> goto T0_init
        fi;
accept_S4:
        if
        :: (! ((q)) && (p)) -> goto accept_S4
        fi;
}
```

PWM Results

1. **Failed** -> Under suitable conditions the H-bridge circuit can be driven to an electrical short.
   - PWM mode is ‘PWMPLS’; duty cycle is less than but near [redacted]%; [redacted] µs task is delayed by ~[redacted] µs; the PWM mode transitions to PWMMNS.
2. **Verified** -> It is provably impossible for inputs to drive the throttle plate always wide open (unless demanded)

NHTSA Findings and Observations

F-10	Extensive software testing and analysis was performed on TMC 2005 Camry L4 source code using static analysis, logic model testing, recursion testing, and worse case execution timing. With the tools utilized during the course of this study, software defects that unilaterally cause a UA were not found.
O-6	While not resulting in a design vulnerability, the MY 2005 Camry source code required unique code inspection tools, and manual inspections due to: a) The TMC software development process uses a proprietary developed coding standard. b) Industry standard static analysis tools provide automated code inspections based upon industry standard code implementations.
O-7	There are no methods for capturing pre-event software state and performance following a UA event either on the vehicle or as a diagnostic tool.

Summary

- Part of the US DOT investigation of Toyota SUA involved analysis of the throttle control software
  - JPL LaRS applied several techniques, including static analysis and logic model checking, to the software
- A handful of logic models were built
  - Some weaknesses were identified; however, no cause for SUA was found
- The full NASA report includes numerous other analyses