

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)

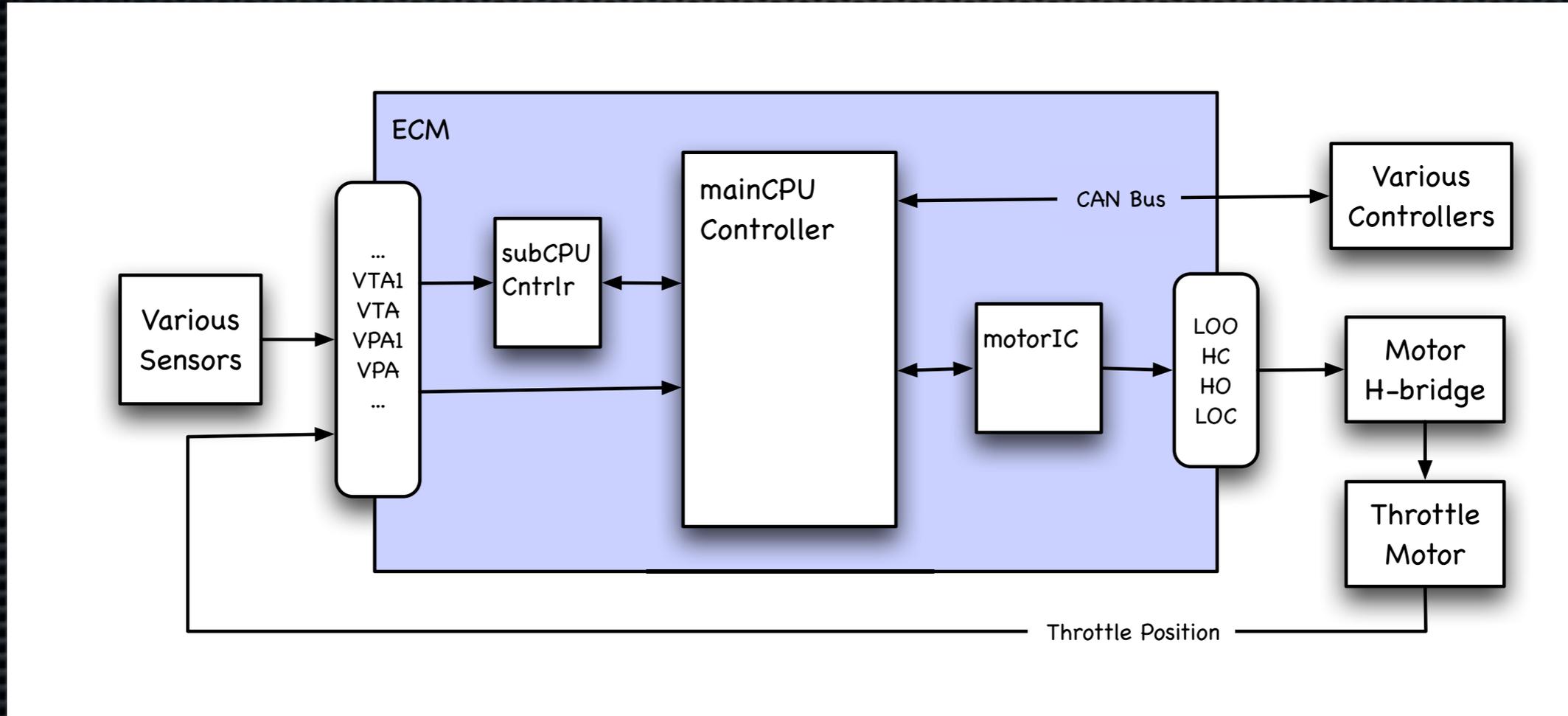
Logic Model Checking (software and systems)

The US DOT, through NASA, established a team and work began - '05 Camry L4

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

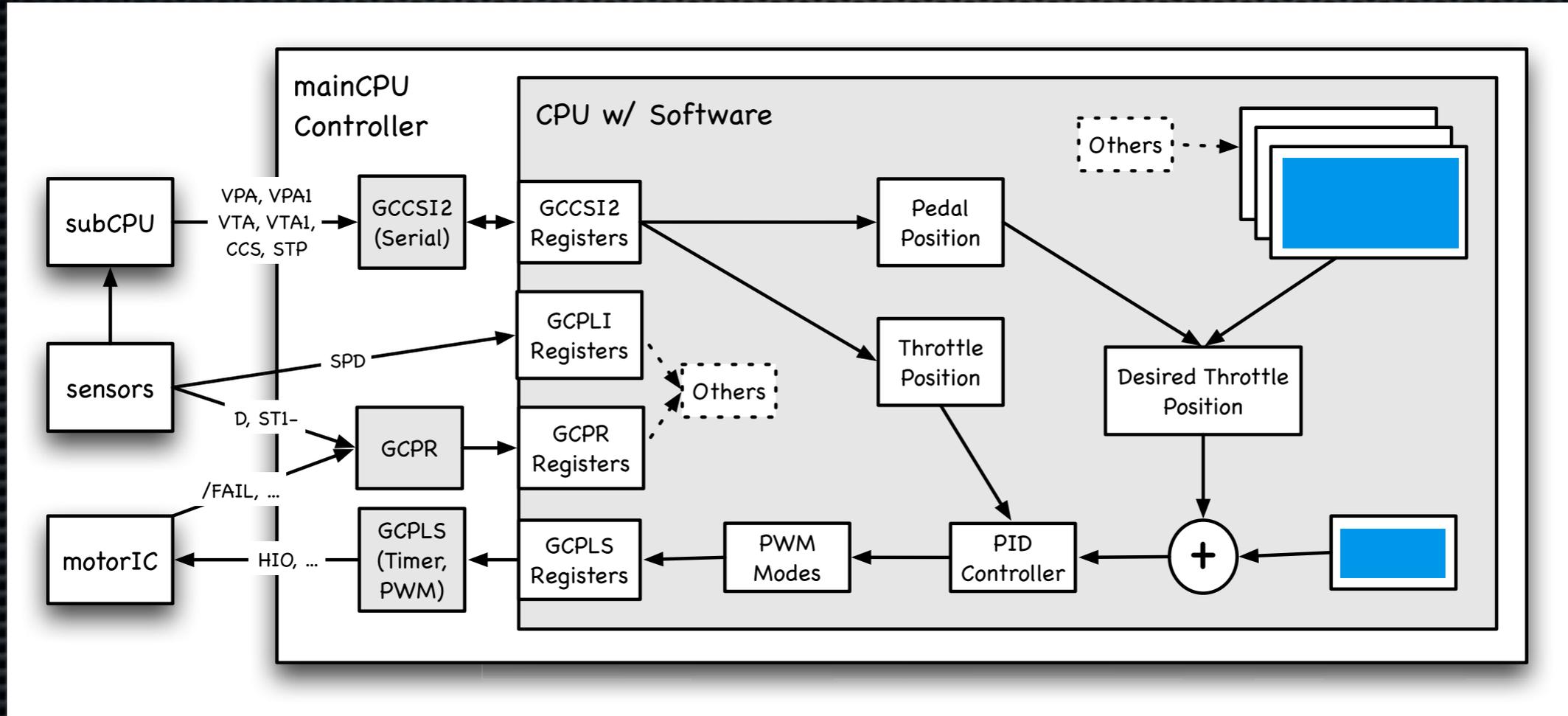
Throttle Control System And Software

Throttle Control System Overview



[Figure 6.4.1-1 - Abstracted]

Throttle Control Software Overview



[Appendix A Figures - Merged]

Throttle Control Software/ System Properties

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

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

LTL Formulae

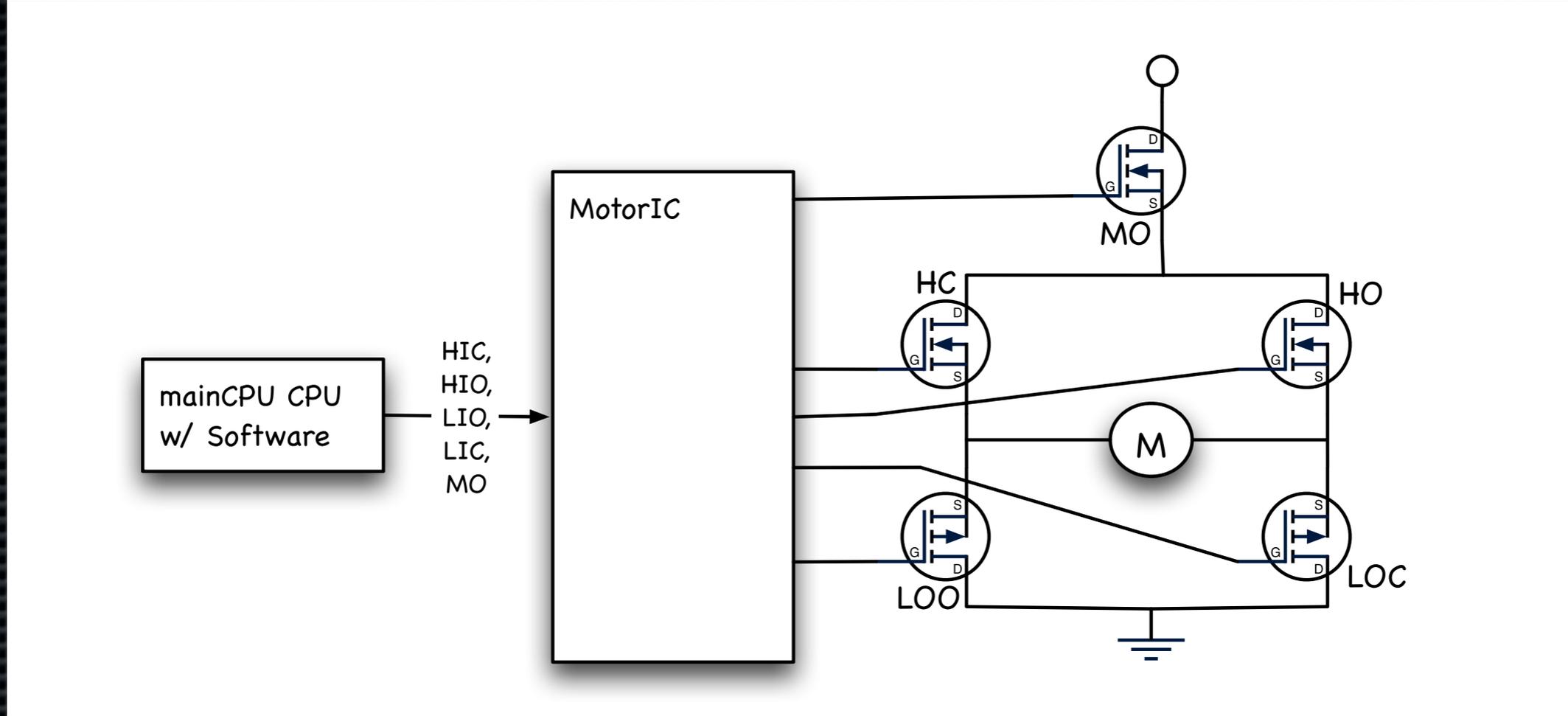
Formula	Reading	Template
$\Box p$	always p	invariance
$\langle \rangle p$	eventually p	guarantee
$p \rightarrow \langle \rangle q$	p implies eventually q	response
$p \rightarrow q \cup r$	p implies q until r	precedence
$\Box \langle \rangle p$	always eventually p	recurrence
$\langle \rangle \Box p$	eventually always p	stability
$\langle \rangle p \rightarrow \langle \rangle q$	eventually p implies eventually q	correlation

Logic Model Analyses

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

Motor Drive IC



Motor Drive IC Correctness

1. The throttle plate is never eventually always wide-open unless the inputs are always demanding wide-open
2. An electrical short never occurs unless the inputs demand that an electrical short occur
3. Verify that all SR-latches are never in their unstable state ($S=1, R=1$ simultaneously)

Motor Drive IC Logic Model

- ✦ Two Promela processes: randomize all inputs and update all outputs.
- ✦ Handle 'over temp' and 'over current' cases
 - ✦ Include 'pre-drive' logic (later version)
- ✦ Assertions for SR-Latch instability

Motor Drive IC

Claim #1 Never Claim

```

/* Motor Direction - ... The variable out_mplus corresponds to the
   throttle plate being driven wide open. ... */
#define out_mplus (out_bm && out_hc && out_loc)
#define out_mminus (out_bm && out_ho && out_loo)

/* CLAIM == 1 - Primary Claim, Requires 'fairness'
 *
 * Show that the throttle plate is never eventually
 * always open when the inputs are not always open. */
#define p out_mplus
#define q (in_hic && in_lic)
never { /* <>[(p && !q) */
T0_init:
    if
    :: (! ((q)) && (p)) -> goto accept_S4
    :: (1) -> goto T0_init
    fi;
accept_S4:
    if
    :: (! ((q)) && (p)) -> goto accept_S4
    fi;
}

```

Motor Drive IC

Claim #1 - Verifier Output

```
./model -a -f -m50000
(never claims generated from LTL formulae are stutter-invariant)

(Spin Version 5.2.5 -- 14 April 2010)
  + Partial Order Reduction
  + Compression

Full statespace search for:
  never claim                +
  assertion violations        + (if within scope of claim)
  acceptance cycles          + (fairness enabled)
  invalid end states         - (disabled by never claim)

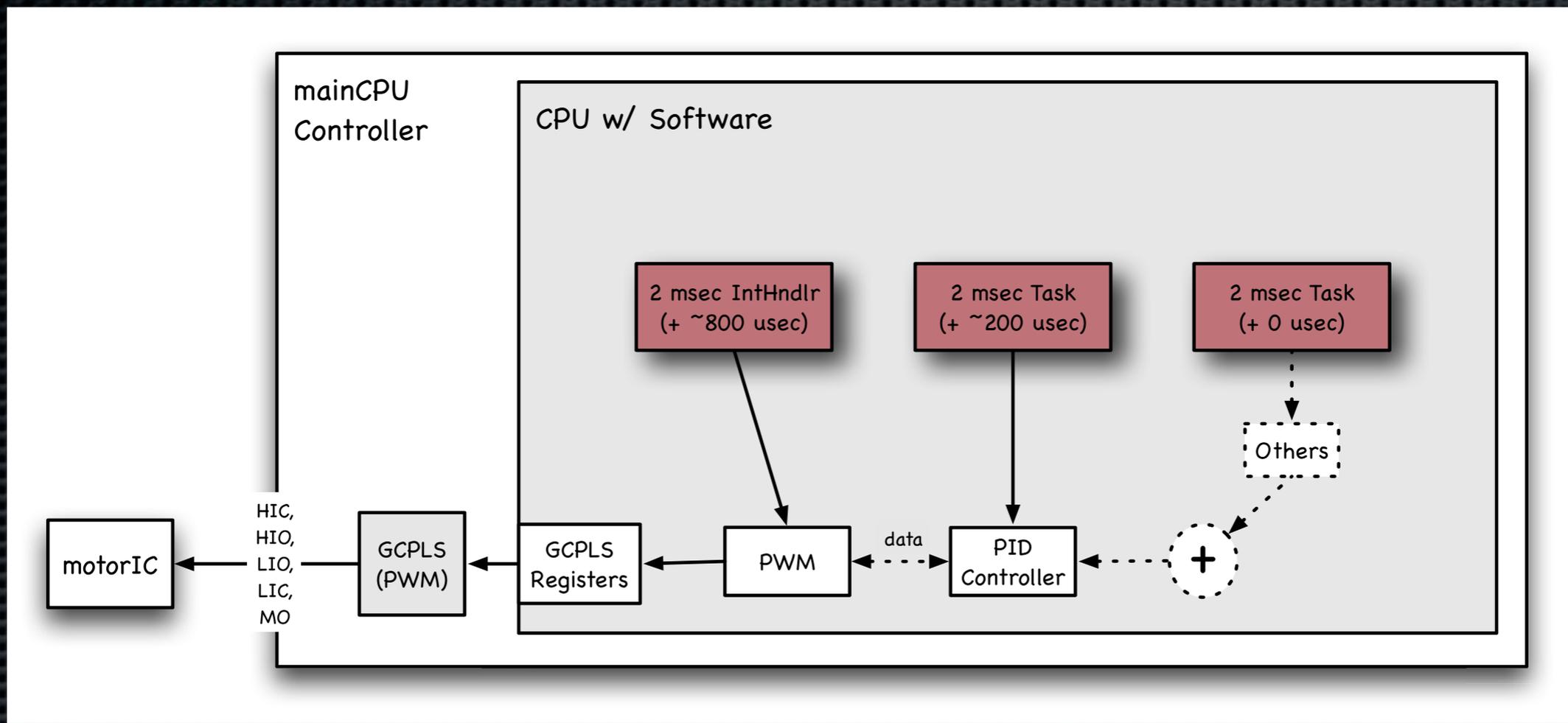
State-vector 28 byte, depth reached 32334, errors: 0
  71790 states, stored (74198 visited)
  1263920 states, matched
  1338118 transitions (= visited+matched)
  384233 atomic steps

hash conflicts:    122808 (resolved)
```

Motor Drive IC Results

1. Verified -> It is provably impossible for inputs to drive the throttle plate wide open (unless demanded).
2. Verified -> It is provably impossible for inputs to drive the H-bridge to an electrical short (unless demanded).
 - ✦ Additional detail : if driven to a short the 'predrive' prevents an H-bridge short.
3. Failed -> SR-latch can be unstable
 - ✦ Additional detail: not actually an SR-latch

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.

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
    :: (!) -> 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
    :: (!) -> 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 █%; █ μ s task is delayed by ~█ μ s; 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

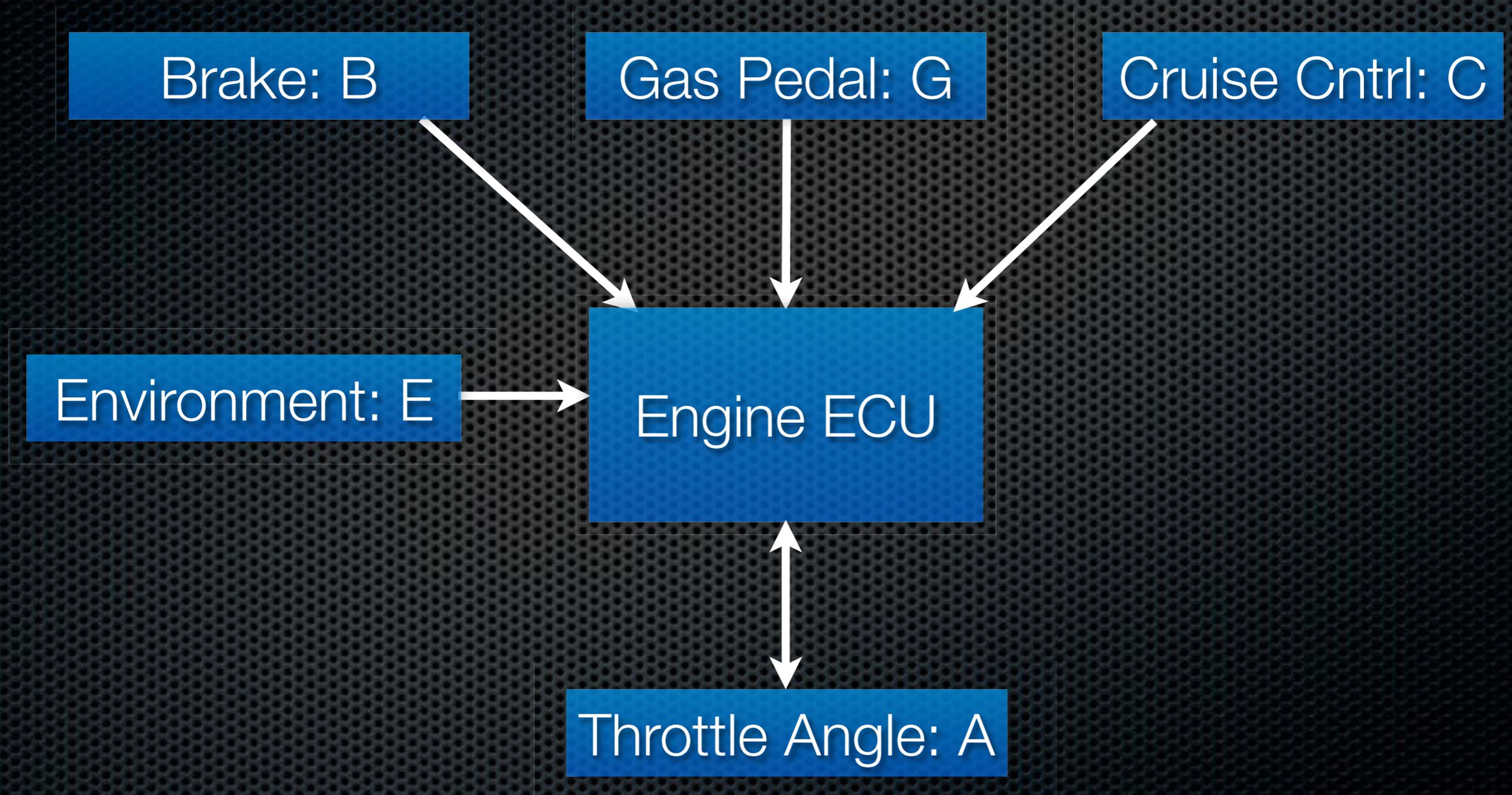
F-10	Extensive software testing and analysis was performed on TMC 2005 Camry L4 source code using static analysis, logic model testing, recursion testing, and worse case execution timing. With the tools utilized during the course of this study, software defects that unilaterally cause a UA were not found.
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: <ul style="list-style-type: none">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

Backup

System Context



Key Assumption

- ✦ A function $A(G,C,B,E,t)$ exists that:
 - ✦ relates the throttle angle over time to all possible pedal, brake, cruise control and environment inputs,
 - ✦ specifies the correct behavior of the vehicle control system.
 - ✦ Definition to come from system design docs.
 - ✦ Simplified, yet reasonable, approximations exist.

Core Correctness Claim

- Every possible temporal combination of inputs (gas pedal, cruise control, others and environment) produces a throttle angle that is consistent with the function: $A(G,C,B,E,t)$.

System Search Approach

- ✦ Exhaustive search for any combination of inputs that leads to a violation of the core system correctness claim.
 - ✦ No constraints placed on ‘Driver’ inputs - any combination, no matter how unrealistic, is possible
 - ✦ No constraints on the environment inputs
- ✦ Specific SUA violation: large throttle angle that is inconsistent (e.g. nonlinear) with the gas pedal position