THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

# Towards Bridging the Computer Science-Control Theory Divide

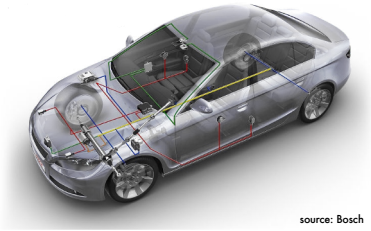
And why

## Certifying Autonomous Systems Software for Modern Architectures is still difficult

**Samarjit Chakraborty**

1

## The Modern Car



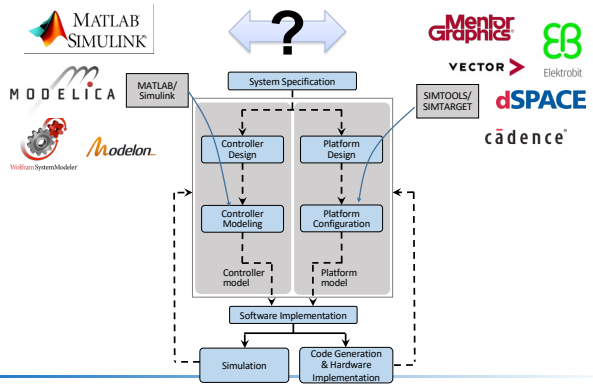
source: Bosch

- 100+ Electronic Control Units (ECUs)
- Complex in-vehicle network (CAN, FlexRay, Ethernet)
- 100+ millions of lines of software code
- ... core of autonomy implemented as distributed control applications (from safety-critical, driver assistance & comfort domains)



2

## Development Workflow

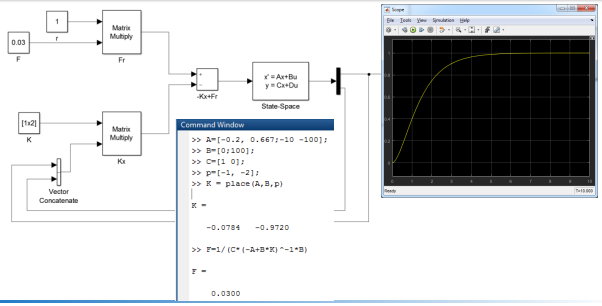





3

## Control Algorithms Design

Controller design and modeling is done in MATLAB/Simulink using closed-loop simulation and analysis

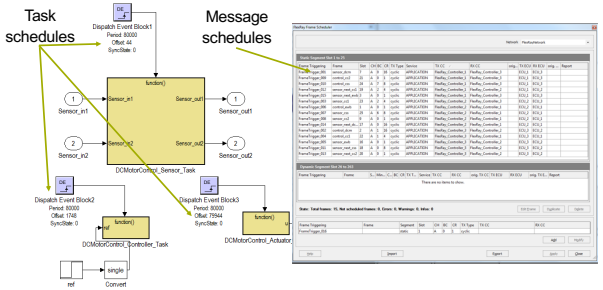





4

## Platform Architecture Design

Platform design and configuration being done using SIMTOOLS/SIMTARGET (hundreds of parameters)

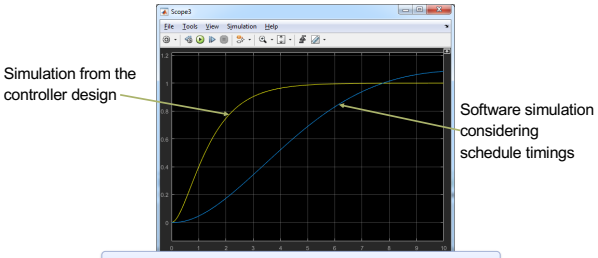





5

## Model versus Implementation

Simulation from the controller design does not match the software simulation




**Why this discrepency?**



6

## Control Systems Design and Implementation

**Equations**

System identification → (system model) → Controller design → (controller) → Control system analysis

**Software**

Code generation → Task partitioning → Task mapping & scheduling → Message scheduling → Timing & performance analysis → Are control objectives satisfied → NO



7

## The Design Flow

**Controller Design**

Control theorist

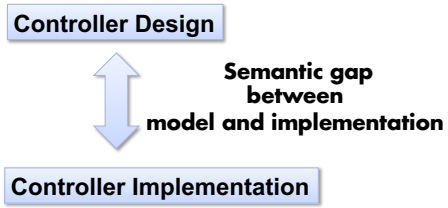
**Design assumptions**
- Computing control law takes negligible time
- No delay from sensor to controller
- No delay from controller to actuator
- No jitter
- …

**Controller Implementation**

Computer engineer

**Implementation reality**
- Tasks have non-negligible execution times
- Often large message delays
- Time and event-triggered communication

## Semantic Gap



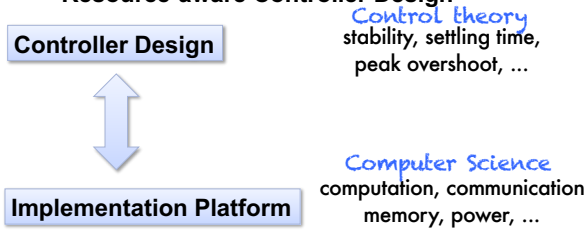

Research Questions?

- How should we quantify this gap?
- How should we close this gap?

9

9

## Resource-aware Controller Design




- Traditionally, Computer Science has been concerned with efficient (implementation of) algorithms - computation, communication, memory, energy, ...
- Metrics for control algorithms have been different ...

10

10

## Computation-aware Design

**Current approach: Safety = Meet all Deadlines**

New approach: Ignore the deadlines,
Focus on what really matters

11

11

## Safety = Meet All Deadlines

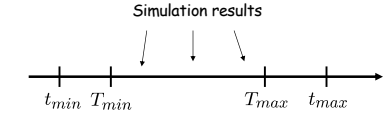
- Multiple controllers on a shared computation resource
- Two stage process:
  - Control engineers design controllers and set deadlines
  - Embedded systems engineers schedule tasks to meet deadlines
- Meeting all the deadlines of the control tasks comes at the expense of pessimistic and inefficient implementations

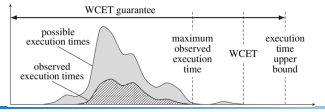
12

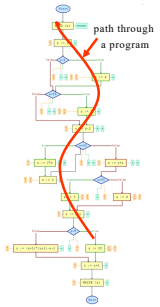
12

## Worst-Case Execution Times of Tasks

- What is the maximum time that a program takes to run on a processor?




13

13

## Safety ≠ Meet All Deadlines

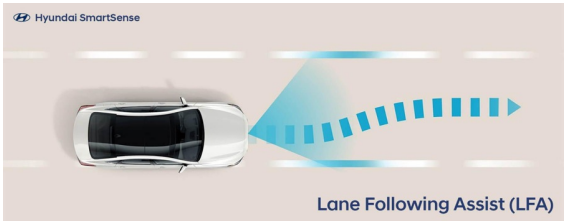
*Can a "system-level" property such as control safety be preserved despite some deadlines being missed?*

14

14

## System-Level Safety

- How do we define safety?
- One notion of safety: the plant deviates from an ideal behavior no more than a predetermined threshold

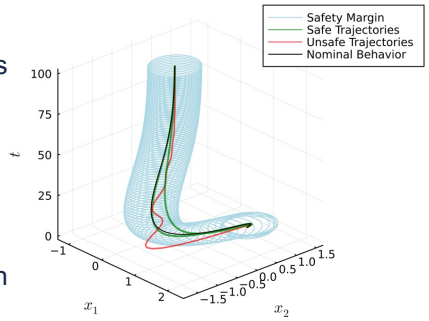



15

15

## Defining Safe Timing Behaviors

- $\binom{m}{k}$: in every window of k periods, at least m deadlines must be met
- Example: $\binom{3}{5}$
  - 1 0 1 1 0 1 0 0 1…
- A timing constraint $\binom{m}{k}$ corresponds to multiple such sequences



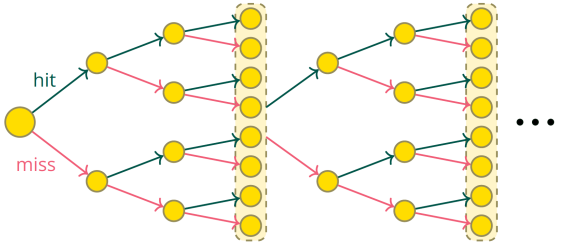

Safe and unsafe system-level behaviors

16

16

## Reachability Analysis

- Over-approximating the reachable by computing a box hull of reachable sets every *r* steps





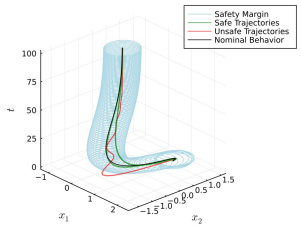

17

## Constraint Synthesis

- A weakly-hard constraint $\binom{m}{k}$ corresponds to a set of trajectories
- $\mathrm{d}(\mathrm{m},\mathrm{k})$: maximum deviation of trajectories that satisfy $\binom{m}{k}$
- We mark constrains with $\mathrm{d}(\mathrm{m},\mathrm{k}) \leq$ safety margin as safe

| Window Size ($k$) | Minimum Hits ($m$) 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 2 | ✓ | – | – | – | – |
| 3 | × | ✓ | – | – | – |
| 4 | × | ✓ | ✓ | – | – |
| 5 | × | ✓ | ✓ | ✓ | – |
| 6 | × | × | ✓ | ✓ | ✓ |

*Safe weakly-hard constraints for Car Suspension (CS)*

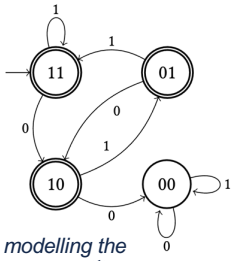




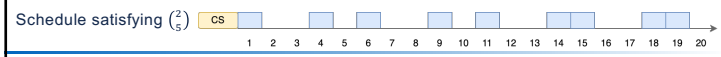

18

## Constraints Constitute a Regular Language

- $\binom{m}{k}$ represents a regular language
- The union of all constraints for a controller is also regular; we call this a controller automaton
- Accepted strings represent safe schedules for one controller




*The automaton modelling the weakly-hard constraint* $\binom{1}{2}$



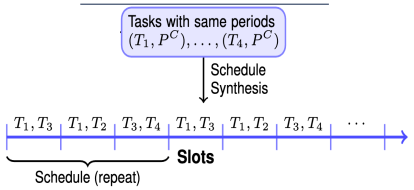



19

## Schedule Synthesis from Regular Languages

- Controller automata → scheduler automaton
- Accepting string represent safe schedules for all controllers
- Interpreting the schedule:
  - Scheduled tasks meet their deadline for that period
  - Non-scheduled tasks miss tl deadline for that period

| Dynamical System | Period |
|---|---|
| RC Network (RC) | 20 ms |
| F1Tenth Car (F1) | 20 ms |
| DC Motor (DC) | 20 ms |
| Car Suspension (CS) | 20 ms |
| Cruise Control (CC) | 20 ms |



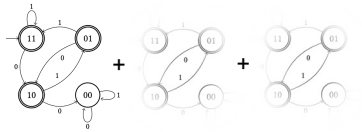


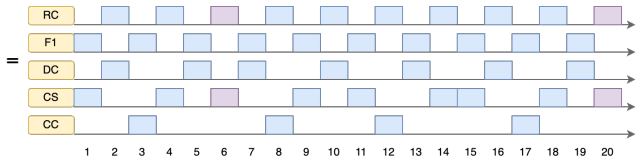

20

## Schedule Synthesis from Regular Languages

- Controller automata → scheduler automaton
- Compose multiple controller automata

| Dynamical System | Period |
|---|---|
| RC Network (RC) | 20 ms |
| F1Tenth Car (F1) | 20 ms |
| DC Motor (DC) | 20 ms |
| Car Suspension (CS) | 20 ms |
| Cruise Control (CC) | 20 ms |

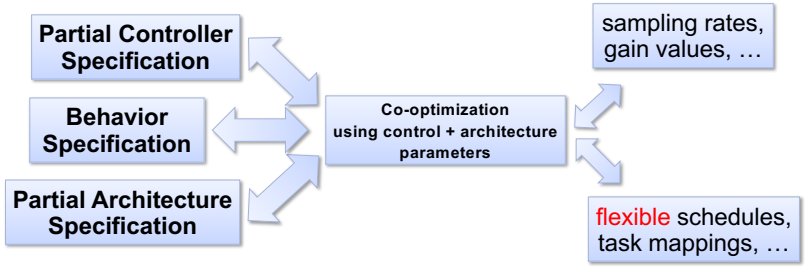





21

## Workflow and Toolchain for Efficient and Certifiable Design




Ongoing NSF project with General Motors & Siemens/Mentor



22