Development of a Production Adaptive Cruise Controller for Heavy Trucks Using Model-Based Design and Production Code Generation

Magnus Eriksson
Adaptive Cruise Control (ACC)
Adaptive Cruise Control

- Driver comfort feature
- Uses complete brake system on truck
  - Exhaust brake
  - Retarder auxiliary brake
  - Foundation brake
- Brake force limited to 3 m/s²
- Driver can select 5 different headway settings
- System available above 20 km/h
Challenges with ACC for heavy trucks

- Greater inertia gives slower systems to control
- Several brake actuators to control
- Big changes in train weights 9 – 60 tons has to be accounted for
- High demands on fuel efficiency
Target Platform

- Scania Gearbox controller
- Freescale MPC563 CPU, 40 MHz
- FPU (floating point support)
- 512 Kb ROM
- 32 Kb RAM
- 32 E²ROM
Platform History

- 1998 First self-made RPS-platform on Scania S6 engine control ECU.
- 2000 ACC prototyping on the improved RPS-platform.
- 2002 The RPS-platform is migrated onto OPC4-hardware (gearbox ECU) and continually upgraded to suite production needs.
- 2006 Adaptive Cruise Control system in production with 100% auto coded application SW
RPS -> Production Platform

- Platform support for production deployment
  - End-of-line Parameterization
  - Diagnostic Trouble Codes
  - Statistics collection
- Modelling guidelines and rules
- Quality assurance of generated code
Platform Principles

- Well defined interface to hand-coded low-level functionality
- One model
  - Simulation
  - Test
  - Code generation
- The model shall always simulate
  - Simulation support for all S-Functions
  - Some of the low level platform code is supported in simulation
- Data is passed in lines or busses
- Information in busses defined in bus objects
Document Generation

Diagnostic Trouble Codes for DAS(AICC)
Simulation Environment

- Small models of specific systems
  - Parameter and controller tuning, e.g. brake performance
- Ability to run controller with data collected in vehicle/HIL
  - Debugging and analysis of truck and HIL tests
- Co simulation with Scania truck simulation library in DYMOLA
  - Mostly used for simulation of fuel consumption
Simulation environment
  - Co-simulation with DYMOLA
    - Application is moved into a specific simulation environment

Truck 2  Truck 1  Target vehicle

Longitudinal controller from Simulink®

Truck model from Dymola
Model Coverage

- Model Coverage metrics was used to evaluate test quality
- The introduction of coverage analysis quickly exposed “dead” and un-testable pieces of model
- Coverage analysis moves the development focus towards testability
- Confidence building
Model vs. Code Coverage

Generated code was run on host with same inputs as the model
- Some code is unaffected by the inputs
- There are some decisions/conditions in code that do not exist in model and vice versa.

Code coverage vs. model coverage
Tool for test vector generation

- A small tool for 100% MC/DC coverage test vectors was developed
  - Based on combinatorial matching of sequences of signals
  - Not strictly Black Box
    - Signal ranges and coarse signal classification has to be supplied
- Accepts all Simulink® and Stateflow® constructs
- Surprisingly effective for this application
Code Generation

- **MISRA Compliance**
  - Do not affect modelling patterns unless considered a valid MISRA violation
  - Manual check and logging of all warnings

- **Code Efficiency**
  - No large-scale comparative study performed
  - Large-scale models believed effective (not proven)
    - Large code modules - abstraction in model not in code
    - High variable reuse, high stack utilization
Comparison between simulation and HIL

- Verify equality of output from ECU with output from simulation environment
  - System with all CAN inputs
  - Deterministic and accurate CAN environment for experiments

- Difficulties
  - State of the ECU at experiment start must be estimated and used in the simulation comparison
  - Exact length of the operations in the ECU are unknown
  - Drift of time base in ECU may differ due to inexact clock
  - Correlation of input/output is unknown (has inputs affected outputs or not)
Comparison between simulation and HIL result

- Combination of manual and automatic estimation/adaptation for
  - State estimation
  - 10 ms “tick” and CAN decoding time
  - Drift of ECU clock
  - Etc… approx 2 man-weeks labour for method and result

- 100 % compliance of outputs verified !!
Conclusions

- Shift of development process
  - Functional development
  - Testing
  - Bench – verification
  - Truck – validation
- Model-Based Design and code generation will increase efficiency and quality
  - Easy to learn
  - Short start times for new project members
  - Faster iterations
  - Higher quality /faster verification
- Auto generated code can be used for production applications