Design and Verification Challenges for Next Generation Automotive Software

S. Ramesh
India Science Lab
GM R&D Labs., Bangalore
Car owners are increasing along with the population increase

- Population: 6B (now) – 7.5B (2020) – 9B (2050)
- From 12% (now) - 15% (2020) - 20% (2050)
- Cars: 7M (now) – 1B (2020) – 1.5B (2050)

1B to 1.5B vehicles is not sustainable!

- Environment
- Energy
- Safety
- Congestion
- Affordability
Customer Requirements

- Energy efficient
- Environmentally friendly
- Safe
- Reliable
- Functional
- Fun to drive
- Affordable

- Stylish
Electronics and SW play a major role in modern vehicles

Introduced a decade ago, it has proliferated the vehicle subsystems
  7000 Ft. of wire length in today’s cars

90% innovation in automobiles is in electronics (Kopetz 2000)

More electronics than in the first airbus
  10s of processors (ECUs), 100s of sensors/actuators
  4-5 different communication buses, 100 millions of lines of code
  10 Mbytes of SW
  % Cost of SW: 1% (1980), 20% (2004), 40% (2015)
Historical Evolution

- Fuel Efficiency: engine and emission control
- Driving Comfort: power steering, ABS, cruise controller, stability
- Safety: belt, airbag controllers, ESP, obstacle detection, driver alerts
- Travel Convenience: ACC, GPS, route planning and navigation aids, multimedia
Future Trends

- Automobiles to Autonomous Vehicles
  - DARPA Grand Urban Challenge
  - GM-CMU is the winner

- Feature enhancement
  - Collision prediction, reduction and prevention
  - Lane, obstacle and occupant aware
  - Driver assist systems, active safety
  - Email, internet, streaming multimedia
  - Communicating vehicles (V2I, V2V)

- Steer-, brake- and throttle- by-wire systems

- Hybrid vehicles

- 360 degrees sensing and integration of functions

- Appropriate HMI
Four diverse categories

- Powertrain control functions
  - Engine control for fuel efficiency
  - Hybrid System, Hard Real Time (micro-, milliseconds)

- Chassis control
  - ABS, ESP, By-wire
  - Hybrid System, Hard Real Time (milliseconds)

- Body electronics
  - Lights, doors, windows, dashboard, seats, mirrors
  - Discrete, Reactive (seconds)

- Telematics
  - Navigation, infotainment (radio, phone, video)
Complex embedded system
- Multiple processors with real-time tasks
- RTOS and middleware: OSEK–RT
- CAN and time-triggered communication buses
- Gateways, routers and protocol stack
- Enormous design and verification challenges
Example of a Backbone Architecture with FlexRay
Computational Features

- Reactive systems
  - Non-termination is a good behavior!
- Hybrid systems
  - Discrete controller for continuous environments
- Distributed systems
  - Irreproducibility of bugs
- Real-time systems
  - Not only right output but at right time
- High degree of reliability
  - Protection from HW failures and SW bugs
  - SW notorious for bugs
- High integrity, safety-critical systems
  - Lack of standards and inspections (unlike avionics)
  - OSI 26262 is just emerging
Design Challenges

- How do we arrive at these products?
  - Correct, reliable and efficient

- Correctness
  - Untrained users, arbitrary environments, large volume

- Reliability and dependability
  - Cost effective and large volume

- Efficiency
  - Hardware resources
  - Software development efforts
Fundamental Conflicts

- Software (discrete) vs. reliability
  - Ariane failure, Therac-25
- Distributed vs. real-time vs. fault-tolerance
  - Time critical in the absence of global clock
- From requirements to production code
  - Requirements are informal, code is formal
- From differential equations to software tasks
  - Different levels of abstractions
- Industrially viable and mathematically rigorous
Current Status

- Time-triggered architectures (Kopetz ’96)
  - TTP, Flexray Buses
- Fault-tolerant middleware (FTCom)
- Real-time operating systems (OSEKTime)
- Model-based development methodologies
  - Simulink/SF, UML, SCADE
- Platform based design
- Component based methodology
  - AUTOSAR
Various tools supporting such methodologies
Commercial and academic
METROPOLIS (Berkeley), SySWeaver (CMU)
STATEMATE, Rhapsody, Object Time (Rational/IBM)
SCADE, Esterel Studio (Esterel Technologies)
dSpace and Mathworks
TTTech, DeComSys Tool Chain
Issues

- Emphasis on the final product or architecture
- Federated SW architecture
  - One or many related functions per ECU/vendor
  - Integration only at communication level and not at functional level
- Multiple methodologies and tools
- Focus on independent single domain rather than at a holistic system level view
- Lack of a single integrated methodology
India Science Lab

- ISL, set up in 2003 in Bangalore
- The only R&D lab. of GM R&D set up outside the NA
- Three major groups
  - Control Software Engineering Methods and Tools Group
  - Vehicular Communication & Info. Management
  - System and SW Architectures
- PhDs and Masters with strong research motivation
- Current Strength around 15
- Would grow to 40 in two years
  - We are looking for people!
- Collaboration with various universities abroad and India
  - CRL with CMU, U Penn, Technion
  - IITs, IISc, TIFR, Honeywell
- Other groups: Manufacturing, Material Science, Vehicle Structures
Taming the Dragon- ISL Approach

- Meta Model Driven Process
- Math based methodology
- Hybrid
- Distributed
- Automotive Software
- Reactive
- Fault tolerant
- Real-time
- High Integrity

India Science Lab
Comprehensive Modeling

- Model Based development
  - Model -> Validate -> Refine -> Auto Code generate
- Modeling all artifacts
  - application control SW, Infrastructure SW,
  - Hardware and Networks
  - Vehicles, Roads and Occupants
- Modeling at different stages
  - Requirements, Algorithms, Design, Code
- Abstract to detailed models
  - For ease of verification and Code generation
- Intuitive but Rigorous
Math-based Approach

- A methodology using precisely defined artifacts at all stages
  - Mathematical semantics and rigorous verification
    - Traditional validation methods inadequate
  - Formal requirements and models
  - Exhaustive verification using symbolic methods
    - Model Checking and Theorem Proving
  - Correctness of refinement leading to consistency of models at different levels
  - Correctness of translation of design models to final code
Math & Model-based Methodology

- Requirement Model
  - Functional Model
  - Env. Model
  - Platform Model
- Mapping & Evaluation
- Code Generation

Formal Verification

India Science Lab
Formal Framework for Correct-by-Construction of Distributed Time Triggered Systems
Network Requirement for the automotive domain
- Higher bandwidth
- Real-Time (Chassis Control applications)
- More reliable operation
  - Deterministic
  - Fault tolerant

Current networks
- CAN is asynchronous and also overloaded
- Safety critical over CAN is VERY complex
Time Triggered Platforms

- Proposed by H. Kopetz
- Emerging like a standard for safety-critical control applications
- Future by-wire platforms likely be DTT platforms
- Options
  - Time Triggered Architecture (TTA) with TTP (TTTech/TTAutomotive)
  - FlexRay (The FlexRay Consortium)
- Multiple distributed nodes with common time frame
- Statically Scheduled Tasks
- Bus based communication
- Communication by TDMA
- Dual redundant bus for fault-tolerance
Design Steps

- Design is very complex and highly iterative
- Functional correctness,
- Timing Correctness: end-to-end constraints
- Para-functional constraints: Fault-tolerance, cost, space

Major Design Steps:
- Development of Functional models (as SL/SF blocks)
- Decomposition of functional model into SW tasks
- Distribution of tasks over different nodes in the TT platform
- Static scheduling of the various tasks
- Message identification and Scheduling
Current Practices & Tools

- TTTech & DeComsys Methodologies
- Major Implementation efforts at GM

Our Observations:
- Highly Manual and error prone
- Adhoc design choices
- Inadequate verification
- Long development cycle
- Person dependent products
What’s difficult?

- Scheduling – especially across OEM <-> supplier relationships
- Ensuring consistency across model transformations
  - Centralized models to distributed implementations
- Para-functionals
  - Signal to frame packing optimization/extensibility
  - Fault tolerance and redundancy
- No simple way to ensure that the final, distributed implementation achieves the same functionality as the centralized, simulated implementation
Where are we?

- Model based methods with auto code generation
  - Some supporting tools
    - Mathworks Matlab Simulink
    - Decomsys tool chain
    - Rhapsody
  - Some internal efforts
    - Body software and controls modeling
    - Powertrain controls modeling

- Focus is on
  - Product lines and separation of behavior from infrastructure
  - Unit testing

- Not a clean slate to start from!
Objectives

- Provide a framework to capture
  - Information from models of control algorithms
  - Constraints on the model transformations
- Semantics of the particular domain/model are implicitly captured

- Consistency across model transformations established by scheduling
  - Static segment of the communication bus
  - Task scheduling on each ECU

- Easy translations from and to existing tool-chains
Centralized Control Model (CCM)

- Cruise Control Subsystem
- Centralized Control Algorithm
  - Instantaneous computation and communication
  - A control algorithm’s point of view

[Diagram of Centralized Control Model (CCM)]

- Actuators
  - Control Signal
  - Vehicle
    - Actuation Signal
    - Wheel Speed
  - Sensors
    - Vehicle Speed

- Cruise Control
  - Desired Vehicle Speed
  - Driver Set Mode

- Signal
  - Speed
  - Wheel

- Centralized Control Algorithm
  - Actuation Signal

- India Science Lab
  - November 29, 2007 • 34
Distributed Control Model

Structural descriptions do not suffice for executing the CCM, we need run time behavior:
- Message schedules (and hence task order)
- Task timing
Centralized Control Model

- A formal model with a clear syntax and semantics
- \( A = \langle S, \langle c, p, \text{offset}_c, \text{deadline}_c \rangle \rangle \)
  - \( S \) – set of blocks
  - \( \langle c \rangle \) – firing order
  - \( P \) – length of the control loop
  - \( \text{Offset}_c \) – earliest firing time of a block
  - \( \text{Deadline}_c \) – latest firing time of a block

- Instantaneous computation and communication
Semantics of CCM

- Sem(A) – captures the firing order of the blocks
- Consists of infinite sequences of certain permutations of the blocks in S
- A permutation X is included provided for all i, j:
  \[ \text{if } X(i) <_c X(j) \text{ and } \text{deadline}(X(i)) < \text{offset}(X(j)) \text{ then } i < j \]

- Semantics allows only those permutations that agree with offset and deadline values.
- Each sequence models a possible execution sequence of the CCM, capturing only the ordering relationship between the blocks.
Class of CCMs

- A is well-formed if the transitive closure of \( <_C \) is irreflexive
  - acyclic control systems - no algebraic loops
- A is consistent if for any block a \( \text{offset}(a) < \text{deadline}(a) \).
- Our focus is on well-formed and consistent CCMs
DCM syntax and semantics

\(<E \cup B, S \cup M, \langle d, \text{distr, wcet, sched, pd}\rangle\)

- E is the set of ECUs
- B is the set of TT buses
- S U M – tasks and messages
- Distr – distribution functions
  - Messages are mapped to buses
- \(\langle d\rangle\) – models the communication relationship
- Sched – begin and end times
- pd – length of the communication cycle

Computation and communication delays
Sem(D) contains infinite sequences of a subset of permutations of S

A permutation $X$ of $S$ is allowed provided, where for each $i, j < |X|,$

- If $\text{end}(X(i)) \leq \text{begin}(X(j))$ then $i < j$
A Class of DCMs

- Well-formed DCM: Every message has a sender and a receiver
- Consistent DCM: begin and end times of tasks are in order and consistent with the data flow relationship
- Non-preempting: tasks allocated to the same nodes are not preempting
  - Can be relaxed
A DCM $D$ correctly implements a CCM $A$, provided
1) $\text{Sem}(D)$ is non empty and a subset of $\text{Sem}(A)$
2) $\text{offset}_c(t) \leq \text{begin}(t) \leq \text{end}(t) \leq \text{deadline}_c(t) \leq p$, for each task $t$ in $S$

These conditions ensure that the data flow and timing relationships between CCM and DCM hold
Main Result

- Suppose CCM A and DCM D are non-preemptive, well-formed and consistent with identical periods
- Then D correctly implements C provided the following conditions hold:
  
  1. Offset(t) <= begin(t) <= end(t) <= deadline(t) <= p for each task t
  2. deadline(t1) < offset(t2) provided t1 and t2 are mapped to communicating tasks in the DCM for each pair of tasks t1 and t2.
Non-preemptive

- \((\text{begin}(\alpha_1), \text{end}(\alpha_1))\) and \((\text{begin}(\alpha_2), \text{end}(\alpha_2))\)
do not overlap \(\forall \alpha_1, \alpha_2\) in \(S\), s.t \(\text{distr}(\alpha_1) = \text{distr}(\alpha_2)\)

Consistent

- \(p_d = p\)
- \(\text{end}(\alpha) = \text{begin}(\alpha) + \text{wcet}(\alpha) \forall \alpha\) in \(S\)
- If \(\alpha_1 <_d \alpha_2\) then \(\text{begin}(\alpha_2) \geq \text{end}(\alpha_1) \forall \alpha_1, \alpha_2 \in (S \cup M)\)

Correct

- \(\text{offset}_c(\tau) \leq \text{begin}(\tau) \leq \text{end}(\tau) \leq \text{deadline}_c(\tau) \leq p\)
  for each task \(\tau\) in \(S\)
- \(\forall \tau_i, \tau_j\) \(\tau_i <_c \tau_j\) and \(\text{deadline}(\tau_i) < \text{offset}_c(\tau_j)\)
  iff \(\tau_i, \tau_j\) are communicating tasks
Correct-by-construction

- Using the constraints and the result stated, we can generate task and message schedules which ensure consistency of the model across the translation from the centralized to distributed implementation.

Verification of existing schedules

- Legacy systems, architectures and processes
  - Introduction of new steps is difficult; hence post verification is easier.

- GM Internal R&D prototype vehicle
  - Prototype vehicle with by-wire braking and steering based on FlexRay.
A few case studies
  - A simple cruise control system
  - Brake-by-wire subsystem

Multi-rate systems
Tens of blocks
Message and task schedule was synthesised for cruise control system
Brake-by-wire subsystem schedule was verified
Given end to end system constraints and signal database

Generate Communication schedule + well formed, non-preemptive, consistent DCM and Solution Sketch Constraints

Generate begin() and end() for all $b_i$

Matlab/Simulink model (with distribution)

Interface Tool

TT Framework model Partial DCM includes distribution, message information

Scheduler 1 (Message schedule)

Scheduler 2 (Message schedule)

... Scheduler n (Message schedule)

TT Framework model Partial DCM + message schedule

Scheduler 1 (Task schedule)

Scheduler 2 (Task schedule)

... Scheduler n (Task schedule)

TT Framework model Complete DCM
Driven by a need to understand and integrate with current day tools for building control applications; introducing light weight, formal processes to augment quality of software produced

Simple approaches often work best; especially within complex work environments and within complex processes

Closer integration with design tools underway

- Interfaces to design tools and schedulers
- Addition of more para-functionals