Develop Vehicle Control Systems As CPS

For Next Generation Automobiles

Shige Wang
GENERAL MOTORS
GLOBAL RESEARCH & DEVELOPMENT
Topics

• Overview of NextGen vehicle control system
  o why is it CPS

• Development process for massive production
  o how is it design and developed: EE system

• Technologies supporting NextGen vehicle control
  o what enables it: real-time embedded systems

• Challenges with new technologies
  o what are missing: parallelism, data processing

3/25/2015  University of North Carolina visit
Cyber-Physical Systems

- **Cyber** – computation, communication, and control that are discrete, logical, and switched
- **Physical** – natural and human-made systems governed by the laws of physics and operating in continuous time
- **Cyber-Physical Systems** – systems in which the cyber and physical systems are tightly integrated at all scales and levels
  - Change from cyber merely appliquéd on physical
  - Change from physical with COTS “computing as parts” mindset
  - Change from ad hoc to grounded, assured development

“CPS will transform how we interact with the physical world just like the Internet transformed how we interact with one another.”

*Source: Dr. Gill presentation at NSF CPS PI meeting*
Controls in Typical Vehicle

- Rear-Passenger Flat-Panel Displays
- Command System with PCMCIA Slot
- GPS Navigation
- DVD Player
- LED Lamp Cluster
- Head-Up Displays
- Dashboard-Instrument Cluster
- Telematic System
- Climate Control
- Electronic Power-Roof System
- Radar Sensor
- Transmission Control
- Collision Avoidance
- Adaptive Cruise Control
- HID Headlamp
- Memory Seat/Mirror/Steer
- Air-Bag Control and Satellite Crash Sensors
- Active Steering
- Body Control
- Suspension Control
- Power Windows
- Remote Keyless Entry
- Seat Massage/HVAC
- Adaptive Brake Lights
- Tire-Pressure-Monitoring
- Parking Sensors
- Rear-View Camera
- Battery Management
- Power Seats
- Throttle Control
- Engine Control Unit
- Folding Door Mirrors
- Car Radio
- Anti-Lock-Braking System/Electronic-Stability Control
- Electrochromic Rear-View Mirrors

3/25/2015
University of North Carolina visit
Vehicle Control System as CPS

- Electrical and electronics replacing mechanic parts
- Standalone system to connected

Take me where I want to go

- Software algorithms + electronic controls and actuators

What’s around me

- 360° sensing (sensors + “V2V”)

Where am I

- GPS + digital maps

Highly complex
- Large number of parts (>2500)
- From different sources
- Built differently
- Last long time (>10 yrs)

Diverse needs
- Expression and self image
- Create personal space
- Market and segment
- Operation environments

Fixed configuration to tailor for different uses

3/25/2015
University of North Carolina visit

- Rely on driver to autonomous driving
- Standalone system to connected

3/25/2015
University of North Carolina visit
CPS Example: Electronic Stability Control

[Diagram showing StabiliTrak ON and OFF scenarios]
With Technology Advancement

Next generation vehicles should be smart and adaptive

- Energy-efficient propulsion
- Vehicle connectivity – both in-vehicle and V2X
- Active safety – driving assistance to autonomy
- Personalized – learn driving styles
- Self management of health
Vehicle Control System: Complexity

- 70+ Electronic Control Units
- 10+ in-vehicle networks
- 100+ sensors and actuators
- 50+ customer features
- 16 domains, 88 subsystems
- Powertrain: 118 functions, 789 signal, 967 links

- 15 concurrent temporal development streams
- 300 hierarchical subsystems
- Thousands of variant features
- Millions of product instances
- Tens-of-thousands of unique product variants

- Highly heterogeneous
- Mixed criticality, mixed intellectual property, mixed versions

- Affordable, reliable, safe, and exciting
- Across a large volume with many variations
- Last long time in all conditions – climate, traffic, maintenance, driving habits, ......
Vehicle Control System Development Process

Multiple stages, multiple groups/organizations, multiple tools, multiple geographical locations

- Development process
  - Requirements
  - Analysis
  - Control Dev
  - SW Design
  - Implementation (RP vs. Target)

- Natural language (Use-case diagram, Block diagram, etc.)
- Diff. Equation, State-based formalism, SDF, etc.
- Continuous + Discrete
- UML, UML Profile, ADL (AADL, EAST-EEA), AutoSAR, etc.
- C/C++ Code Libraries

- Programming Env (Tornado, MS Visual Studio, etc.)
- dSpace Target Link
- Rhapsody
- Matlab SL/SF

- Model translation/composition
- Tool integration

- SW Design
  - DOORS
  - Excel
  - Word

- Implementation (RP vs. Target)
  - C/C++ Code Libraries

- Regional Architecture

- Lifecycle Processes
  - Concept of Operations
  - System Requirements
  - High Level Design
  - Detailed Design
  - Software/Hardware Development
  - Field Installation
  - System Verification Plan
  - System Verification
  - System Validation Plan

- Timeline

- Conceptualization and Definition
  - Feasibility Study/Concept Exploration

- System Validation Plan
  - System Validation

- Retirement/Replacement

- Operations and Maintenance
  - Changes and Upgrades

- MD Chart

- Vehicle Control System Development Process

- University of North Carolina visit

3/25/2015
**Vehicle Control System Engineering**

### Functional Architecture

**Integration Area**
- Production Line Models
- Customer Requirements*

**Domain Area**
- Logical View
- Production Line Models
- Product Requirements*

### Implementation Architecture

**SW/HW Component Compositions**
- Algorithm Models & Calibration Definition

### Deployment Architecture

**System Design Model**
- SWC
- HWC
- Networks
- ECU

**ECU Extracts**
- Controller Development
- Core Arch Design
- Diagnostic Design
- Network Design

### Vehicle Application Architecture

- Hardware Connectivity
- Power & Ground
- Merge with 3D data for wiring
- Service Documentation

### System Data

- Change Management
- Integration Platform

3/25/2015
University of North Carolina visit
**Challenge: Software Implementation of Controls**

- **Different design principles and focuses**
  - Code generation in commercial tools limited to small component
- **Timing delays cause errors**
  - Mismatch implementation and model
  - Introduce timing jitters
  - Non-determinism across different configurations
- **Multitasking and multirate control**
  - Determine proper software tasks for control
  - Determine proper schedule of software tasks
- **More challenging – parallel programming and execution**
  - Resource sharing, data protection, synchronization, etc
There may be a correlation between sampling period and execution time.

Per = 10, wc completion time = 1~5

Per = 6, wc completion time = 1~5
ABS Example: Physical Variability Impact
ABS Example: Software Implementation Impact
Standardization: AUTOSAR

**SW-C**
(application software components)

**VFB**
(virtual functional bus)

**RTE**
(run-time environment)

**BSW-M**
(basic software modules)
Components interact through ports:

- **Sender-receiver:** SWC-SWC, 1-many, async
- **Client-server:** SWC-BSW, many-1, sync

For **Sender-Receiver ports:**
- Initial value
- Queue length
- Explicit vs. implicit
- Acknowledgement
- Timeout

For **Client-Server ports:**
- Synchronous vs. asynchronous
- Timeout
- Queue length
**SW Components and Runnables**

- **SW-Components**
  - atomic block with respect to *mapping*
  - provided by one supplier

- **Runnables**
  - atomic block with respect to *execution*
  - attach to different OS tasks
Example: Front-Light Management (Distributed)
AUTOSAR OS

Basic feature
- Configured and scaled statically
- Amenable to reasoning of real-time performance
- Provides a priority-based scheduling policy
- Provides protective functions at run-time (for memory, time, etc)
- Can run on low-end controllers and without external resources

ECU State Manager
- starts/stops AUTOSAR OS

Interaction with RTE
- Map runnables of the same SW component to task(s)
- Share protection boundary among runnables of the same task
- Tasks and ISRs for basic SW scheduled by OS

OS Abstraction Layer
- Define for co-existence of AUTOS OS and proprietary OS
**AUTOSAR Scheduling**

- Two level, table-driven
- Used for static scheduling: all tasks are synchronized with alarms
  - Alarms fixed once started
  - Each table linked to one tick counter
- Defines expiry points
- Allow multiple table activated concurrently – compositional schedule
- Two types of schedule tables: one-shot, repeated
AUTOSAR OS Configuration

```
#define START_TIME_10MS 0
#define CYCLE_TIME_10MS 10

COUNTER SYSTEM_COUNTER {
    /* 1ms system counter */
    MAXALLOWEDVALUE = 65535;
    TICKSPERBASE = 1000000;
    M NCYCLE = 1;
};

ALARM <MODULEPREFIX>_TRANSMIT_ALARM {
    COUNTER = SYSTEM_COUNTER;
    ACTION = ACTIVATE_TASK {
        TASK = <ModulePrefix>_MainFunction_TransmitTask;
    };}:
```

```c
#include "<ModulePrefix>.h"
#include "SchM.h"
#include "SchM_cfg.h"
#include "SchM_<ModulePrefix>.h"

void
SchM_Init () {
    SetRelAlarm (<MODULEPREFIX>_TRANSMIT_ALARM,
                 START_TIME_10MS,
                 CYCLE_TIME_10MS);
}

TASK (<ModulePrefix>_MainFunction_TransmitTask) {
    <ModulePrefix>_MainFunction_Transmit ();
    TerminateTask ();
}
```
OS Application

- A set of OS objects to form a functional unit
- Can be trusted or non-trusted
- Objects of the same OS application can access each other
Deployment View

ECU 1 – Lane keeping ECU
- Left Sensor Component
- Lane Keeping Application Component
- AUTOSAR Stack

ECU 2 – Sensor Components
- Right Sensor Component
- Left Sensor Component
- AUTOSAR Stack

ECU 3 – Displays & Controllers
- Display and Controller components
- AUTOSAR Stack

Legend
- Application Components
- ECU
- Basic Software Components
- Flex Ray Bus

Flex Ray Bus

3/25/2015 University of North Carolina visit
AUTOSAR Fault-Tolerant: E2E Protection

- Mitigate faults in E2E communication
- Defined a set of E2E profiles – non-generated, deterministic code
- E2E library defines 3 Profiles
# E2E Communication Profiles

<table>
<thead>
<tr>
<th>Errors</th>
<th>State of the art mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hardware Redundancy</td>
</tr>
<tr>
<td>Deletion</td>
<td>x</td>
</tr>
<tr>
<td>Repetition</td>
<td>x</td>
</tr>
<tr>
<td>Timing delay</td>
<td>x</td>
</tr>
<tr>
<td>Incorrect Sequence</td>
<td>x</td>
</tr>
<tr>
<td>Insertion of unintended data</td>
<td>x</td>
</tr>
<tr>
<td>Data Corruption</td>
<td>x</td>
</tr>
<tr>
<td>Addressing Error</td>
<td>x</td>
</tr>
<tr>
<td>Masquerading</td>
<td>x</td>
</tr>
<tr>
<td>Inconsistency</td>
<td>x</td>
</tr>
<tr>
<td>Constant &quot;over-&quot; Transmission</td>
<td>x</td>
</tr>
</tbody>
</table>

3/25/2015  University of North Carolina visit
Safe E2E Communications

- **Receiver 1**
  - Application Logic
  - E2E protection wrapper
  - Receive safe data elements
  - Invoke safe read do get the data element
  - Call E2E protection wrapper

- **Sender**
  - Application Logic
  - E2E protection wrapper
  - Produce safe data elements
  - Invoke safe transmission request
  - Call E2E protection wrapper

- **E2ELib**
  - Call E2E check on array

- **AUTOSAR Runtime Environment (RTE)**
  - RTE communication (intra or inter ECU), either through COM, IOC, or local in RTE
  - RTE_Read_<p>_<o>() to get the data element
  - RTE_Write_<p>_<o>() to transmit the data element

- **OS-Application 1**
  - Consumer of safe data elements

- **OS-Application 2**
  - Producer of safe data elements

University of North Carolina visit
Memory Protection

• Use OS-Applications: two variants

- OS-App 1 private data
- OS-App 2 private data
- OS-App n private data
- OS-App 1 private code
- OS-App 2 private code
- OS-App n private code
- Optional shared OS-App 1 data (buffer used by RTE for IPC)

- OS-App 1 private data
- OS-App 2 private data
- OS-App n private data
- OS-App 1 private code
- OS-App 2 private code
- OS-App n private code
- Optional shared OS-App 1 data (buffer used by RTE for IPC)

- OS-Application 1, trusted, with protection enabled
- OS-Application 1, trusted, with protection disabled
Timing Protection

• Support specifying and verifying timing properties (period, latency, jitter, synchronization, execution time, deadline)

• Two implementation: BSW time service; RTE sync event

ECU 1
OS Task, ECU 1
Local Time Tick
synchronized global time

ECU 2
OS Task, ECU 2
Local Time Tick
synchronized global time

ECU 1
synchronized RTE events

ECU 2
synchronized RTE events

SW-C
RTETimingEvent

SW-C
RTETimingEvent

SW-C
RTETimingEvent

time sync protocol

sync event
Program Flow Monitoring

- Focuses on SWC program flow faults
- Two types: timely behavior; sequence of code blocks
- Use SWC ports and Watchdog Manager

![Diagram](image_url)
## AUTOSAR OS Scalability Class

<table>
<thead>
<tr>
<th>Feature</th>
<th>Scalability Class 1</th>
<th>Scalability Class 2</th>
<th>Scalability Class 3</th>
<th>Scalability Class 4</th>
<th>Hardware requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSEK OS (all conformance classes)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Counter Interface</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>SWFRT Interface</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Schedule Tables</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Stack Monitoring</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>ProtectionHook</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Timing Protection</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Timer(s) with high priority interrupt</td>
</tr>
<tr>
<td>Global Time /Synchronization Support</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>Global time source</td>
</tr>
<tr>
<td>Memory Protection</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>MPU</td>
</tr>
<tr>
<td>OS-Applications</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Service Protection</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>CallTrustedFunction</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>(Non-)privileged Modes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature</th>
<th>Scalability Class 1</th>
<th>Scalability Class 2</th>
<th>Scalability Class 3</th>
<th>Scalability Class 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum number of Schedule Tables supported</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>Minimum number of OS-Applications supported</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Minimum number of software Counters supported</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
AUTOSAR Multicore OS

Assumptions
- Individual cores can be identified
- Cores share instruction set and data size
- Exceptions stay within a core
- Interrupts can be triggered on any core
- Equal access of memory (or segment)
- No addition of cores after OS started, no shutdown/restart of individual cores
- No dynamic task assignment

Core organization: master - satellite
- Master: all BSW, wakeup/shutdown management
- Satellite: subset BSW only for CDD and SW-Cs

ECU State Manager on each core
- Synchronize startup, shutdown, sleep operation
- A single configuration of OS for all cores
- Each core runs a part or whole copy of OS
AUTOSAR Considered Multicore SW Architectures

- History: some options considered

- Same OS on all cores
- One OS handles all cores
- One OS per core
- OS only on one core, other cores are black boxes

- Single Image MP
- Dynamic Affinity MP
- Static Affinity MP
- Multi Instance MP
- Multi-processing

- RTE Bridged MP
- Inter-Connection on AR RTE level
- Interrupt off-loading

- COM Bridged MP
- Inter-Connection on AR COM level
- Distributed calculations

- I/O Processing
- Co-Processing

- CPU Farm

Tasks and IRQs fixed to cores
**AUTOSAR Multicore OS Scheduling**

- Fixed priority-based schedule on each core
- Cores execute task independently
  - Each core has its own schedule table(s)
- Task priorities on different cores are limited to local
**Multicore OS Startup**

- Each core must have at least one OS Application
- Master-satellite can be cascade
- All cores start before StartOS command
Multicore Timing Control

- Each core must have at least one counter from timer
- Multiple counters exist for OS Applications
- Counters can be on remote cores
- Synchronization among counters may be required
- Implementation is not specified
Multicore Inter OS-Application Communicator (IOC)

- For communications between OS-Applications across cores or memory protection boundary
  - In addition to Intra OS (via RTE) and Inter ECU (via COM)
  - Accessed directly (not support) or through RTE (logic one, physical splitted)
  - 1:1 or N:1 (1:N requires RTE generating N 1:1, changed in 4.2+)

3/25/2015

University of North Carolina visit
Mutual Exclusion Across Cores

- **Spinlocks**
  - A busy-waiting mechanism polling the variables until available
  - Used by tasks on different cores
  - Priority-based scheduling on each core not affected
- **Deadlock avoidance**
  - Resolved using predefined access order: all cores follow the same order
  - Controlled by LIFO policy
**New Technology: Advanced Driver Assistance System (ADAS)**

- **Human Interface Strategy**
  - Motorized Seat Belts
  - Collision Mitigation Braking
  - Rear Back-Up Camera
    - Human Interface Strategy
      - Aux Virtual Image Display (AVID)
- **Front Camera Functions**
  - Lane Departure Warning
  - Forward Collision Alert
- **Side Blind Zone Alert**
- **Adaptive Cruise / Full-Speed Range ACC**
- **Collision Mitigation Braking**
- **Motorized Seat Belts**
- **Haptic Seat**
**Computing Challenges for ADAS**

- **Very high computation demands**
  - Image processing, complex math computation
  - Fusing and cross-examining information from different sources

- **Very large amount of data**
  - Fast and continuous sampling
  - Many streams of inputs

- **Under real-time and embedded constraints**
  - Source of information for critical controls (speed, steering, braking, etc)
  - Key part of system safety (ASIL-D), but with a weak computing platform support (ASIL-B)

- **Other business challenges**
**Hardware Options**

**IMX 6Quad Applications Processor Block Diagram**

- **System Control**
  - Secure JTAG
  - PLL, Osc.
  - Clock and Reset
  - Smart DMA
  - IOMUX
- **Power Management**
  - Voltage Monitor
  - Internal Memory
    - ROM
    - RAM
- **Security**
  - RNG
  - TrustZone
  - Secure RTOS
  - Cryptography
- **Display and Camera Interface**
  - HDMI and PHY
  - 24-bit RGB, LVDS (x4)
  - MIPI DSI
  - MIPI CSI2
- **Connectivity**
  - USB 2.0 OTG and PHY
  - USB3 Host and PHY
  - 2GB OTG and PHY
  - LV-CSI
  - MIPI CSI2
  - MIPI DSI
  - MIPI SDI
- **Video Codec**
  - SD
  - Audio
  - Audio
- **Image Processing Unit**
  - Video Compression/Decompression
  - Power Management
  - Temperature Monitor

**Expansion Port (DP/LVDS, Touch SPI, I2C, CSI x1, CSI x4, GPIO 6.0)**

**Features**

- **Main MCU**
  - Image Recognition Engine
  - Graphic Engine
  - Disparity Compensation Module
- **System Power Supply IC**
  - ADAS IC
- **Video Input**
  - NTSC ADC
  - Display Control
- **Dynamic Range Control**
- **Video Output**
  - HDMI
  - SDI
Software Architecture Challenge: mixed-criticality

• Classified for each feature
  o Forward collision avoidance – ASIL D
  o Lane keeping – ASIL C
  o Lane departure warning – ASIL A

• Different ASIL for functions in the same feature
  o Object detection – ASIL B
  o Video streaming – ASIL A

• ASIL may change in different modes during operation
  o Back up: rear camera for monitoring and warning – ASIL C/D
  o Normal driving: rear camera for monitoring – ASIL B

3/25/2015

University of North Carolina visit
Software Architecture Challenge: virtualization

- Consider hypervisor
- Mixed different OSs to maintain independence of applications
- Different impl. strategies
  - Pass-through/native virtualization
  - Para-virtualization
- Some examples
  - Research: KVM, Xen, OKL4
  - Commercial: QNX, WindRiver, COQUOS, PikeOS, GH IntegrityOS, Coqos
Software Architecture Challenge: memory management

- **Performance depends more on memory**
  - Same for multicore and GPU
  - Memory protection doesn't help

- **Data locality should be explored**
  - Cache, system memory, device memory
  - Partition of memory into continuous regions

- **Cache- and memory-locking**

---

3/25/2015

University of North Carolina visit
Programming

• Programming Model
  o Data parallelism
  o Task parallelism

• Languages
  o OpenCL
  o CUDA
  o OpenMP
Other Issues

• Task synchronization
• Data coherence and consistency
• Processor power consumption and temperature
Remarks

• Software development methods for large scale software for mass produced CPS are still not mature
• New applications post additional technical and business challenges
• Advanced hardware offers great potentials but also posts challenges
• Software architecture supporting predictable and analyzable vehicle controls is still evolving and requires further research
Interference Among Controls

- Algorithms designed to run standalone
- Interferences of other controls through resource share causing more inconsistency (temporal and data)

Each line represents activation of a distinct control (a schedule)
- Running standalone, they all should be with regular pulses
- Diagram shows running on a shared controller without any protection (code-gen, flash, run)
- Observed extensive blocking (no pulse) and busy-processing (piggyback pulses)
Simulation Setup – Architecture Selection

- Test Scenario
  - Host vehicle starts from 0 with 200 ft behind lead vehicle
  - Lead vehicle moves with a predefined speed profile

- Task running synthetic workload
  - Tasks are not randomly generated (like most academic cases)
  - To be filled with control algorithm after it is done

- Processor utilization should be maintained less than 70%
  - For single core, implies sampling period 80 ms and more
Performance With Single or Dual-Core

40 ms

80 ms
Performance With Different Periods

Running on a resource restricted platform, pick 800 ms
If preserve control performance, pick 60 ms
Performance With Different Synchronization

40 ms lock-free, wait-free

40 ms lock-based
Performance Under Different Sync and Period