From Electronic Design Automation to Automotive Design Automation

Chung-Wei Lin
cwlin@csie.ntu.edu.tw
Assistant Professor
CSIE Department
National Taiwan University
April 2019
Connected and Autonomous Vehicles

- A good application may need both of "connectivity" and "autonomy"

- What if the intersection management does not have connectivity?
- What if the intersection management does not have autonomy?
Connected Applications

- Connectivity realizes more applications, together with
  - ADAS
  - Autonomous functions

- Intersection Management
- Emergency Vehicle Warning
- Jamming
- Spoofing
- Cooperative Adaptive Cruise Control
- Side Road Merging
- Sharp Curve Assistant
Software Design Complexity

- Various applications including Advanced Driver Assistance Systems (ADAS) and autonomous functions

- Various software programs for sensing, signal processing, control, decision making, etc.
  - Embedded software value to vehicle's total value
    - 2% → 13% from 2000 to 2010
  - Number of lines of code
    - 1 → 10+ → 100 million from 2000 → 2010 → now

- Due to the safety-critical nature, correctness and quality of software are extremely important

http://www.toyota.com/safety-sense/
Hardware Design Complexity

- **Number of Electronic Control Units (ECUs)**
  - 20 → 50+ in the past decade

- **Integrated architecture**
  - One function can be distributed over multiple ECUs, and multiple functions can be supported by one ECU
    - More sharing and contention among software functions
    - Traditional federated architecture: each function is deployed to one ECU and provided as a black-box by its supplier

- **New computational components**
  - Field Programmable Gate Array (FPGA)
  - Graphical Processing Unit (GPU)

- **Next-generation communication protocols**
  - Ethernet-based protocols

https://en.wikipedia.org/wiki/Ethernet
"Design Automation"

- Consider different design metrics
  - Safety, reliability, robustness, performance, etc.
- Assist system designers for early design decisions

https://en.wikipedia.org/wiki/V-Model_(software_development)
EDA vs. Automotive Design Automation

Modeling

- Architecture functional model
- Periodic activation
- Schedule

Design

- Functional model
- Periodic activation

Analysis

- Input
- Output
- Periodic activation

Electronic Design Automation (EDA)

- Intensity induced by aperture p
- Intensity induced by aperture q

Automotive Design

- Current path from A to VCC3A
- Current path from VCC to B
EDA: Wire Routing and Wire Sizing
Similar Problem in Automotive Design

- The wiring weight of a system can be up to 30kg
  - The third heaviest and costliest component in an automotive system (after the chassis and the engine)
- Netlist
  - A set of "parts" to be connected
- Splice
  - Used for connecting more than two wires
  - Steiner vertex!
- Where to put splices?
  - Steiner tree problem
EDA: FPGA and Bio-Chip Routing

- **FPGA routing**

- **Bio-chip routing**

Similar Problem in Automotive Design

No Traffic Light + No Communication

Traffic Light 5s

Traffic Light 10s

No Traffic Light + Communication

The Histogram of the Delay Time for Traffic in 10min (Average Arrival Interval 4s)

Extension to Multiple Lanes
# Outline

<table>
<thead>
<tr>
<th>#1</th>
<th>#2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Placement</td>
<td>Verification</td>
</tr>
<tr>
<td>#3</td>
<td>#4</td>
</tr>
<tr>
<td>Software Integrity</td>
<td>Security-Aware Design and Analysis</td>
</tr>
</tbody>
</table>
One Example Formulation

- Software (functional model): task graph
- Hardware (architectural platform): distributed Electronic Control Units (ECUs) connected by a network
One Example Solution

- Decide task allocation and assign priorities to tasks on ECUs and messages on the network
- Satisfy timing constraints for tasks, signals, and paths
Edge Computing (1/2)

From Automotive Edge Computing Consortium
Edge Computing (2/2)

From Automotive Edge Computing Consortium
<table>
<thead>
<tr>
<th>#1</th>
<th>#2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Placement</td>
<td>Verification</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>#3</td>
<td>#4</td>
</tr>
<tr>
<td>Software Integrity</td>
<td>Security-Aware Design and Analysis</td>
</tr>
</tbody>
</table>
The traditional system development process is the V-model

- An OEM defines the specifications of components
- Suppliers implement those components

Formal verification can be applied to design models and implementations

- However, its scalability limits its applicability to systems of high complexity

Runtime monitoring becomes a practical alternative

- Detect and notify when there is any specification or requirement violation during runtime
Case Study

- Integration of two systems
  - Cooperative Pile-up Mitigation System (CPMS)
  - False-start Prevention System (FPS)

- Property specification language and automation tool
  - Signal Temporal Logic (STL)
    - Extend Linear Temporal Logic (LTL) to specify properties over real time
  - Breach [Donze '10]
    - Given a STL formula, synthesize an online monitor as a C++ program or a MATLAB S-function which can be realized as a Simulink block

- An assumption violation of CPMS is detected!
Outline

#1 Placement
#2 Verification
#3 Software Integrity
#4 Security-Aware Design and Analysis
ISO 26262

- ISO 26262 is recognized as the state-of-the-art standard for functional safety of automotive systems

Examples

- Some software structures are NOT recommended for highest Safety Integrity Level (SIL)
  - Dynamic objects and variables
  - Multiple uses of variable names
  - Implicit type conversions
  - Unconditional jumps
  - Recursions
Motivations

- A potential conflict between certification issuers (e.g., OEM) and software suppliers (developers)
  - A certification process represents a systematic way to inspect the source codes
  - Some source codes of software suppliers (developers) are confidential

- Desired properties
  - Authenticity
    - Only authenticated results from compilers and analysis tools (verification, simulation, and/or testing) are considered by the certification issuers
  - Confidentiality
    - Sensitive source codes of the software suppliers and developers are not released to certification issuers
Certification Protocol

- **Trusted third-party**
  - Run a certification program which consists of a compiler and an analyzer
  - Maintain a router which controls the input and the output

- **Certification program**
  - All of the compiler, the analyzer, and the private key are updated by the OEM
  - The updating process must be unidirectional to guarantee confidentiality

- **Router**
  - Only the corresponding developer can be the receiver
## Outline

<table>
<thead>
<tr>
<th>#1 Placement</th>
<th>#2 Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#3 Software Integrity</th>
<th>#4 Security-Aware Design and Analysis</th>
</tr>
</thead>
</table>
"Design Automation"

- Consider different design metrics
  - Safety, robustness, performance, security, etc.

- Assist system designers for early design decisions
  - More efficient process
Security-Aware Design and Analysis

- Security is a rising concern, especially with connectivity

- One hypothetical (but very likely) scenario
  - Design stage
    - Use the RSA algorithm (strong and famous) for encryption, decryption, and authentication!
  - Implementation stage
    - Computing units on vehicles cannot afford it... (security mechanisms are usually computation-intensive)
  - Result: redesign systems (how can we prevent this?)
Cooperative Adaptive Cruise Control (CACC)

- **Two CACC modes**
  - **Gap control mode**
    - The following vehicle (F) decides acceleration based on the gap, speeds, and accelerations of the two vehicles
  - **Collision avoidance mode**
    - The following vehicle (F) decelerates with its maximum deceleration

- **Information sources**
  - Gap and speeds are obtained by sensors
  - Accelerations are broadcasted with V2X messages
CACC with Jamming or Lying

Car 1: 3 m/s
Car 2: 1 m/s, 8 m
Car 3: 0 m/s, 6 m
Intersection Management
(With Jamming or Lying)

- An intersection manager receives requests from vehicles, schedule them, and sends confirmations to them
Intersection Management
(Payment-Based Solution against Lying)

☐ The payment-based approach supports prioritized intersection management where truthfulness is guaranteed

☐ An intersection becomes "more expensive" when there are more cars requesting the intersection
## Summary

<table>
<thead>
<tr>
<th>#1</th>
<th>#2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Placement</td>
<td>Verification</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>#3</td>
<td>#4</td>
</tr>
<tr>
<td>Software Integrity</td>
<td>Security-Aware Design and Analysis</td>
</tr>
</tbody>
</table>
Q&A

Thank You!