Can SYCL and OpenCL meet the challenges of functional safety

April 9th 2021

Illya Rudkin, Principle Engineer - Lead Safety Practitioner
Aim of this presentation

Look at some of the challenges put to adopting SYCL and OpenCL for safety critical applications

• Give you a context of ISO 26262
• How functional safety affects the SYCL stack
• Example architecture of an automotive platform
• How the SYCL stack today supports safety and some aspects that could be improved
• Quick history of autonomous vehicles
Aim of this presentation

Functional safety is a big topic and so this presentation will not cover:

- The RTOS
- The tool chain and tool qualification
- Statistics / results / reports
- Online offline kernel compilation and execution
- Khronos® safety critical specifications
- Comparison with other parallel computing platforms like CUDA®
History of autonomous vehicles

- 1500
- 1868
- 1925
- 1933
- 1939
- 1945
- 1950
- 1969
- 1977
- 1987
- 1995
- 2002
- 2003
- 2009
- 2013
- 2018
- 2020
- 2021

1985
1972
1990
2000
2002
2003
2009
2013
2018
2020
2021

ISO 26262
ISO 21448
SOTIF

Codeplay®
© 2021 Codeplay Software Ltd
History of autonomous vehicles

Source: United States Department of Transport's NHTSA
What can the industry do about it?

Create a safety standard
Automotive Functional Safety Standard ISO 26262

First published in 2011, and the current second edition was released in 2018.

- Covers all electrical systems in a vehicle
- Widely adopted by most automotive OEMs
Fact: Complex software contains bugs and edge cases
Fact: Not all bugs will be found or fixed

Fact: Most bugs or edges cases can be mitigated with **good software engineering practices**, the analysis of the requirements backed up with a **verification strategies** appropriate to the consequences of failure.
Eliminate of Software Bugs

There are generally two types of bugs:

- Random failure
- Systematic failure
  - Human error
  - Human misunderstanding or assumptions
  - Limited testing coverage to catch ‘non-conformities’
Automotive Functional Safety Standard ISO 26262

Brief:
• Covers all electrical systems in a vehicle
• Widely adopted by most automotive OEMs
• It consists of two parts:
  • Builds on the ISO 9001 quality management standard
  • Additional safety activities
How does ISO 26262 achieve this?

Through Analysis + Verification
Quality Managed Software Development - ISO 9001

- Product Requirements
- Product Design
- History of design development
- Component Implementation
- Product Test
- Product Released
- Verification
- Project Management
- Configuration Management
- Requirements Management
- Quality Management

Engineering Activity
Management Activity
Automotive Functional Safety Standard ISO 26262

- Component Design
- Component Req. Analysis
- System Design
- System Req. Analysis
- Component Implementation
- Component Test
- Component Integration
- System Integration
- System Test
- Product Released
- Product Requirements
- Verification

- Project Management
- Configuration Management
- Requirements Management
- Quality Management

- Engineering Activity
- Management Activity
- ISO 26262 Activity
ISO 26262

The management of risk

Factors:
- Severity - the potential of harm
- Exposure – the probability of occurrence
- Controllability – the ability to avoid injury

Requirements + Analysis
Typical Automotive ASIL Classifications

ASIL = Automotive Safety Integrity Level

Airbag: Inadvertent deployment
ASIL D

Instrument Cluster: Loss of critical data
ASIL B

Vision ADAS: Incorrect sensor feedback
ASIL A

Rear Lights: Failure both sides
ASIL A

Rear View Cameras: No valid sensor data
ASIL B

Brake Lights: Failure both sides
ASIL B

Engine Management: Unwanted Acceleration
ASIL C & D

Active Suspension: Suspension Oscillates
ASIL B & C

Headlights: Failure Both Sides
ASIL B

Radar Cruise Control: Inadvertent Breaking
ASIL C

Antilock Braking: Unintended full power braking
ASIL D

Electric Power Steering: Self Steering
ASIL D

ASIL = Automotive Safety Integrity Level

Image credit: Mentor
ISO 26262 – It is about risk of failure management

Determining the ASIL levels for Electrical Systems

Malfunction: Brake ABS system failure

Hazard Analysis: State the unintended situations (hazards) that could occur

Risk Analysis: How likely is the hazard to happen?

ASIL Determination: What level of safety (risk) reduction does the system need?

ASIL = Automotive Safety Integrity Level
ISO 26262

Get to a safe state

Reach your safety goal requirement
ISO 26262:2018 - Fault Tolerant Time Interval (FTTI)

“The minimum timespan from the occurrence of a fault in an item to a possible occurrence of a hazardous event, if the safety mechanisms are not activated.”

Stack reached safe state

Safety mechanism can be:
• A limited operation mode
• Output an error status
• Shutdown
• Reset
• Activate safety systems
• Deactivate until within operational parameters
• An external operational monitor

Fault occurs

Malfunctioning behavior

Hazard occurs
(Safety goal violated)

Hazard manifestation period

Malfunction manifestation period

-time
ISO 26262

Key word: Deterministic

Strives for everything to be understood and confirmed
What is not ‘deterministic’ in the stack?

- Thread dead lock
- Kernel execution time
- Interrupt kernel execution
- Denial of Service call
- Dynamic memory latency
- Execution performance
- Data integrity
- Memory read write order
- Auto optimized code
- Concurrency
- Application resource management
- Freedom from interference
- Resource availability
- C++ exception handling
- Scheduler
- Compiler
- Rogue kernels
- Code execution paths

Can you provide assurances these will not cause a failure?
The automotive SYCL compute stack
A SYCL automotive compute stack is very much like typical automotive compute stacks that exist today.
A SYCL compute stack is very much like automotive compute stacks that exist today.

**Sensory or communications platform**
- Vehicle CAN
- Radar
- GPS
- V2X
- Vision camera
- LIDAR

**ADAS or Autonomous**
**SYCL application**
- Prediction
- Planning
- Object detection
- Sensor Fusion
- Mapping
- Driver monitoring

**RTOS**
- SYCL library
- Backend driver e.g. OpenCL library
- Device driver(s)

**Heterogenous platform desired device(s)**
- CPU
- FPGA
- GPU
- DSP
- AI ASIC

**Safety Mechanism**
- Safe state reached = Safety goal met
- Detailed error or status propagation

**On error**
So, what is ISO 26262 asking you to do?

1st By removing the issues

We reduce the chance of failure by
- Having clear requirements
- Performing analysis
- Verification of the implementation

2nd On a failure occurring

Initiate the safety mechanism

→ It runs you through a set of design and software requirements
ISO 26262 Part 6: Software Development

As a developer you would have to demonstrate:

• How you would carry out the activity
• Validate the activity
• Produce evidence of the activity
The ISO 26262 requirements – The levels of rigor to apply

Table 1 — Topics to be covered by modelling and coding guidelines

<table>
<thead>
<tr>
<th>Topics</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>1a Enforcement of low complexity(^a)</td>
<td>++</td>
</tr>
<tr>
<td>1b Use of language subsets(^b)</td>
<td>++</td>
</tr>
<tr>
<td>1c Enforcement of strong typing(^c)</td>
<td>++</td>
</tr>
<tr>
<td>1d Use of defensive implementation techniques(^d)</td>
<td>+</td>
</tr>
<tr>
<td>1e Use of well-trusted design principles(^e)</td>
<td>+</td>
</tr>
<tr>
<td>1f Use of unambiguous graphical representation</td>
<td>+</td>
</tr>
<tr>
<td>1g Use of style guides</td>
<td>+</td>
</tr>
<tr>
<td>1h Use of naming conventions</td>
<td>++</td>
</tr>
<tr>
<td>1i Concurrency aspects(^f)</td>
<td>+</td>
</tr>
</tbody>
</table>

\(^a\) = need not apply
\(^b\) = recommended practices or ‘should be using anyway’
\(^c\) = highly recommended practices or ‘mandatory’. A justification is needed if not used.
The ISO 26262 requirements – The levels of rigor to apply

Table 3 — Principles for software architectural design

<table>
<thead>
<tr>
<th>Principles</th>
<th>ASIL A</th>
<th>ASIL B</th>
<th>ASIL C</th>
<th>ASIL D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a  Appropriate hierarchical structure of the software components</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1b  Restricted size and complexity of software components(^a)</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1c  Restricted size of interfaces(^a)</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>1d  Strong cohesion within each software component(^b)</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1e  Loose coupling between software components(^b,c)</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1f  Appropriate scheduling properties</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1g  Restricted use of interrupts(^a,d)</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>1h  Appropriate spatial isolation of the software components</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>1i  Appropriate management of shared resources(^e)</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
</tbody>
</table>

\(^a\) = o = need not apply
\(^b\) = + = recommended practices or ‘should be using anyway’
\(^c\) = ++ = highly recommended practices or ‘mandatory’. A justification is needed if not used.

ASIL = Automotive Safety Integrity Level
# The ISO 26262 requirements – The levels of rigor to apply

ASIL = Automotive Safety Integrity Level

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Walk-through of the design\textsuperscript{a}</td>
<td>++</td>
</tr>
<tr>
<td>1b Inspection of the design\textsuperscript{a}</td>
<td>+</td>
</tr>
<tr>
<td>1c Simulation of dynamic behaviour of the design</td>
<td>+</td>
</tr>
<tr>
<td>1d Prototype generation</td>
<td>0</td>
</tr>
<tr>
<td>1e Formal verification</td>
<td>0</td>
</tr>
<tr>
<td>1f Control flow analysis\textsuperscript{b}</td>
<td>+</td>
</tr>
<tr>
<td>1g Data flow analysis\textsuperscript{b}</td>
<td>+</td>
</tr>
<tr>
<td>1h Scheduling analysis</td>
<td>+</td>
</tr>
</tbody>
</table>

\textsuperscript{a} o = need not apply

\textsuperscript{b} + = recommended practices or ‘should be using anyway’

++ = highly recommended practices or ‘mandatory’. A justification is needed if not used.
The ISO 26262 requirements – The levels of rigor to apply

### Table 6 — Design principles for software unit design and implementation

<table>
<thead>
<tr>
<th>Principle</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>1a  One entry and one exit point in subprograms and functions(^a)</td>
<td>++</td>
</tr>
<tr>
<td>1b  No dynamic objects or variables, or else online test during their creation(^a)</td>
<td>+</td>
</tr>
<tr>
<td>1c  Initialization of variables</td>
<td>++</td>
</tr>
<tr>
<td>1d  No multiple use of variable names(^a)</td>
<td>++</td>
</tr>
<tr>
<td>1e  Avoid global variables or else justify their usage(^a)</td>
<td>+</td>
</tr>
<tr>
<td>1f  Restricted use of pointers(^a)</td>
<td>+</td>
</tr>
<tr>
<td>1g  No implicit type conversions(^a)</td>
<td>+</td>
</tr>
<tr>
<td>1h  No hidden data flow or control flow</td>
<td>+</td>
</tr>
<tr>
<td>1i  No unconditional jumps(^a)</td>
<td>++</td>
</tr>
<tr>
<td>1j  No recursions</td>
<td>+</td>
</tr>
</tbody>
</table>

ASIL = Automotive Safety Integrity Level

\(^a\) = need not apply

\(+= \text{recommended practices or ‘should be using anyway’}\)

\(++= \text{highly recommended practices or ‘mandatory’. A justification is needed if not used.}\)
# The ISO 26262 requirements – The levels of rigor to apply

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL A</th>
<th>ASIL B</th>
<th>ASIL C</th>
<th>ASIL D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Walk-through</td>
<td>++</td>
<td></td>
<td>o</td>
<td>0</td>
</tr>
<tr>
<td>1b Pair-programming</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>1c Inspection</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1d Semi-formal verification</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1e Formal verification</td>
<td>o</td>
<td>0</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>1f Control flow analysis</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1g Data flow analysis</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1h Static code analysis</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1i Static analyses based on abstract interpretation</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>1j Requirements-based test</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1k Interface test</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>1l Fault injection test</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>1m Resource usage evaluation</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>1n Back-to-back comparison test between model and code, if applicable</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
</tbody>
</table>

**ASIL = Automotive Safety Integrity Level**

- **o** = need not apply
- **+** = recommended practices or ‘should be using anyway’
- **++** = highly recommended practices or ‘mandatory’. A justification is needed if not used.
## The ISO 26262 requirements – The levels of rigor to apply

ASIL = Automotive Safety Integrity Level

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Analysis of requirements</td>
<td>++</td>
</tr>
<tr>
<td>1b Generation and analysis of equivalence classes&lt;sup&gt;a&lt;/sup&gt;</td>
<td>++</td>
</tr>
<tr>
<td>1c Analysis of boundary values&lt;sup&gt;b&lt;/sup&gt;</td>
<td>++</td>
</tr>
<tr>
<td>1d Error guessing based on knowledge or experience&lt;sup&gt;c&lt;/sup&gt;</td>
<td>++</td>
</tr>
</tbody>
</table>

<sup>a</sup> = need not apply, <sup>b</sup> = recommended practices or ‘should be using anyway’, <sup>c</sup> = highly recommended practices or ‘mandatory’. A justification is needed if not used.

---

<sup>a</sup> = need not apply, <sup>b</sup> = recommended practices or ‘should be using anyway’, <sup>c</sup> = highly recommended practices or ‘mandatory’. A justification is needed if not used.
## The ISO 26262 requirements – The levels of rigor to apply

### Table 10 — Methods for verification of software integration

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a Requirements-based test&lt;sup&gt;a&lt;/sup&gt;</td>
<td>++</td>
</tr>
<tr>
<td>1b Interface test</td>
<td>++</td>
</tr>
<tr>
<td>1c Fault injection test&lt;sup&gt;b&lt;/sup&gt;</td>
<td>+</td>
</tr>
<tr>
<td>1d Resource usage evaluation&lt;sup&gt;c, d&lt;/sup&gt;</td>
<td>++</td>
</tr>
<tr>
<td>1e Back-to-back comparison test between model and code, if applicable&lt;sup&gt;e&lt;/sup&gt;</td>
<td>+</td>
</tr>
<tr>
<td>1f Verification of the control flow and data flow</td>
<td>+</td>
</tr>
<tr>
<td>1g Static code analysis&lt;sup&gt;f&lt;/sup&gt;</td>
<td>++</td>
</tr>
<tr>
<td>1h Static analyses based on abstract interpretation&lt;sup&gt;g&lt;/sup&gt;</td>
<td>+</td>
</tr>
</tbody>
</table>

**Notes:**
- **ASIL** = Automotive Safety Integrity Level
- **o** = need not apply
- **+** = recommended practices or ‘should be using anyway’
- **++** = highly recommended practices or ‘mandatory’. A justification is needed if not used.
The ISO 26262 requirements – The levels of rigor to apply

Table 15 — Methods for deriving test cases for the test of the embedded software

<table>
<thead>
<tr>
<th>Methods</th>
<th>ASIL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>1a Analysis of requirements</td>
<td>++</td>
</tr>
<tr>
<td>1b Generation and analysis of equivalence classes</td>
<td>+</td>
</tr>
<tr>
<td>1c Analysis of boundary values</td>
<td>+</td>
</tr>
<tr>
<td>1d Error guessing based on knowledge or experience</td>
<td>+</td>
</tr>
<tr>
<td>1e Analysis of functional dependencies</td>
<td>+</td>
</tr>
<tr>
<td>1f Analysis of operational use cases(^\text{a})</td>
<td>+</td>
</tr>
</tbody>
</table>

\(^\text{a}\) = need not apply

\(+\) = recommended practices or ‘should be using anyway’

\((++\) = highly recommended practices or ‘mandatory’. A justification is needed if not used.

ASIL = Automotive Safety Integrity Level
Conclusion

SYCL and OpenCL specifications and current implementations are not designed with safety in mind.

With regards to the SYCL application stack meeting the challenge of ISO 26262, it has:
• Good integration design principles
  • Separation of concerns
  • Clearly identifiable API boundaries, call hierarchy and responsibilities
  • Supports integration testing
• High quality Khronos specifications with requirements

To support safety mechanisms and deterministic behavior throughout the stack, it needs:
• Resource management models and verification methods
• Concurrency and scheduling of task models
• Improved support for communicating propagating up status and error conditions
• Improved support for negative testing
• Data integrity checking
• Operational self test support and reporting
Thank you

Contact: illya@codeplay.com

Enable AI and HPC to be open, safe and accessible to all