SYCL SC State of the Union
IWOCL’24

April 10, 2024
Victor Perez & Hugh Delaney
On behalf of the SYCL SC WG
Agenda

- **Background**
- **Highlights of last 12 months**
- **Ecosystem**
Khronos Safety Critical Standards Evolution

Khronos has 20 years experience in standards for safety-critical markets
Leveraging proven mainstream standards with shipping implementations and developer tooling and familiarity
A choice of abstraction levels to suit different markets and developer needs

- **OpenGL ES 1.0 - 2003**
  - Fixed function graphics

- **OpenGL ES 2.0 - 2007**
  - Programmable Shaders

- **OpenGL SC 1.0 - 2005**
  - Fixed function graphics safety-critical subset

- **OpenGL SC 2.0 - 2016**
  - Programmable Shaders Safety-critical subset

- **Vulkan 1.2 - 2020**
  - Explicit Graphics and Compute and Display

- **Vulkan SC 1.0 - 2022**
  - Explicit Graphics, Compute and Display safety-critical subset

- **OpenVX SC Extension – 2017**
  - Graph-based vision and inferencing

- **OpenVX 1.3 - 2019**
  - SC Extension integrated into core OpenVX specification

- **OpenVX 1.3 - 2019**
  - SC Extension integrated into core OpenVX specification

- **SYCL 2020**
  - C++-based heterogeneous parallel programming

**SYCL SC Working Group created to develop C++-based heterogeneous parallel compute programming framework for safety-critical systems**

**March 2023**

© The Khronos® Group Inc. 2024 - Page 3
SYCL SC Working Group Officers

Verena Beckham  
VP of Safety Engineering  
at  
Codeplay

Chair

Leonidas Kosmidis  
Senior Researcher  
at  
BSC

Spec Editor

Andriy Byzhynar  
Software Architect  
at  
Intellias

Outreach Officer
SYCL SC Working Group Regular Members

intellias
codeplay
CoreAVI
arm
SYCL SC
Mercedes-Benz
Collins Aerospace
AMD
Qualcomm
RENESSAS
mobileye
What is “Safety-Critical”?

- A system is *Safety-Critical* if its failure could result in harm/death of people
- **SC industries:** automotive, avionics, medical, rail, atomic
- **Often certified according to standards**
  - Automotive: ISO 26262
  - Avionics: DO-178C
  - Medical: IEC 62304
- **Standards define safety levels:** ASIL A-D / DAL A-E / Class A-C
- **Require Functional Safety**
  - Absence of unreasonable risk caused by malfunction
  => Risk has been analyzed, mitigated to a reasonable level, proven
  - A system property
  - More than just language safety
SYCL SC

• Why?
  - SC industries increasingly require *acceleration* of software, due to
    - Rising popularity of AI algorithms
    - Proliferation of *heterogeneous* computing
    - Increasing demand for *performance*

• What?
  - Based on SYCL 2020
  - Modifications to ease safety-certification
    - Of the implementation of the standard
    - Of the SYCL application
What SYCL SC is Not

SYCL SC will not

- Tell you how to implement a “safe” application
- Guarantee a safe application
- Tell you how to implement a “safe” SYCL SC runtime
- Guarantee a safe runtime
- Tell you how to apply any industry process or standard
- Be certified (as a standard)
- Make your hardware safe

SYCL SC will be compatible with you doing the above, but cannot do it for you.
SYCL SC assumes that you are using safe HW, e.g. incorporating redundancy, EDC/ECC, watchdogs.
Agenda

- Background
- **Highlights of last 12 months**
- Ecosystem
Online Compilation

- A SYCL implementation can do one or both of:
  - Online compilation of kernels at run-time
  - Offline compilation of kernels
- Some SYCL features rely on online compilation, e.g.
  - Specialization constants
  - Parts of kernel_bundle

- All deployed SW needs to be safety certified
- Safety certification is expensive
  - Follow strict processes
  - Write code in a careful way (e.g. follow guidelines)
  - Perform exhaustive testing
  - ...
- Don’t want to certify a compiler!
- Offline compilation allows verification of binary during development phase

Implies: Focus on offline compilation only in SYCL SC.
Development vs. Deployment

Traditional SW development

Development → Deployment

Safety critical SW development

Development → Certification → Deployment

This work is licensed under a Creative Commons Attribution 4.0 International License
Development vs. Deployment Features

Examples:

<table>
<thead>
<tr>
<th>Development Feature</th>
<th>Deployment Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>profiling support</td>
<td>queues</td>
</tr>
<tr>
<td>stream class</td>
<td>buffers/accessors</td>
</tr>
</tbody>
</table>

⚠️ All deployed code must be safety certified

💰 Certification is expensive

Remove development features ➡ Lower certification costs

Development features can still be implemented outside of the spec!

Expect:
- Debug & Release builds of SYCL SC runtime OR
- Use SYCL implementation for development, move to SYCL SC for deployment

Implies: Make it easy for a SYCL SC application to run on a SYCL runtime
Additional Discussion Items

The WG has also started discussing

Avoiding Dynamic Memory on Host
- Dynamic memory in C++ is not deterministic
- All memory allocation typically static or up-front in SC applications
- Finding a balance between determinism and algorithm flexibility

Deterministic Error Management
- SYCL uses C++ exception
- Timing of exception handling not deterministic in common compilers
- Some custom compilers support this
- Challenge: Keep the difference to Base SYCL small
Outreach

Safety Critical Open Standards for Accelerated Heterogeneous Computing

We have seen an explosion in Machine Learning and AI solutions over the past decade due in part to the ecosystem of open standard libraries and frameworks that enable engineers to prototype ideas quickly. Now, as the need increases for safety-critical APIs that can meet application engineers at levels of abstraction that they are familiar with, open standards for high-level abstraction in safety-critical heterogeneous computing such as SYCL Safety Critical and those that facilitate low-level access to GPU acceleration for advanced graphics and compute applications, such as Vulkan Safety Critical are enabling applications in safety-critical markets such as automotive, avionics, industrial, and medical. This session will also discuss how OpenVX provides a safety profile for deploying discrete vision algorithms and Neural Network inferencing. This session explores how these safety-critical standards adhere to MISRA C++ guidelines and align with safety-critical standards such as such as RTCA DO-178C Level A / EASA ED-12C (avionics), ISO 26262/21443 (automotive), IEC 61508 (industrial), and IEC 62304 (Medical).

In addition to member presentations
Agenda

- Background
- Highlights of last 12 months
- Ecosystem
Unified Acceleration Foundation (UXL)

Mission

- Build a **multi-architecture multi-vendor software ecosystem** for all accelerators
- **Unify** the heterogeneous compute ecosystem **around open standards**
- Build on and expand **open source projects for accelerated computing**

oneAPI Specification

- **oneDPL**
  - Data Parallel C++ Library
- **oneDNN**
  - Deep Neural Network Library
- **oneCCL**
  - Collective Communications Library
- **oneDAL**
  - Data Analytics Library
- **oneTBB**
  - Threading Building Blocks
- **oneMKL**
  - Math Kernel Library

www.uxlfoundation.org
New Safety Critical SIG

Aim: Enable/accelerate integration of UXL projects into safety critical systems

Potential activities:

- Analyse/Suggest changes to make projects easier to safety certify;
- Communicate SC-specific requirements;
- Discuss certification/integration strategies;
- Collaborate on SYCL SC porting & safety artefacts.

Open to anyone

To join: https://lists.uxlfoundation.org/g/Safety-Critical-SIG
Khronos AUTOSAR Liaison: SYCL Demonstrator

Motivation

Currently there is no native AUTOSAR functionality to utilize hardware accelerators for high performance computation. Only way is to integrate 3rd party libraries which can affect safety.

At the same time there is a challenge for AUTOSAR Adaptive Platform to cover cutting-edge functionality like:
- AD/ADAS systems
- Performing heavy algorithms
- AI
- etc.

The main aim: creation of generic API in AUTOSAR, which allows to utilize hardware acceleration for computation efficiency improvement. SYCL is the best candidate to be used under the hood. Moreover, SYCL SC will potentially add required safety compatibility.

Thank you to AUTOSAR and Intellias

The main goal of this concept is to enable parallel heterogeneous programming, using standardized C++ based API, for solving issue of high performance computing.

Important part of the concept is to consider ISO-26262 Standard without sacrificing of performance.
Get Involved!

Excited about getting your hands on this?
Are you piqued by the challenge?

Get in contact!
Member of Khronos? Join the Working Group!
Not a member? Look out for Advisory Panels!

Visit www.khronos.org/syclsc
Contact sycl_sc-chair@lists.khronos.org
or verena@codeplay.com