Acknowledgement and Disclaimer

Numerous people internal and external to the original C++/Khronos group, in industry and academia, have made contributions, influenced ideas, written part of this presentations, and offered feedbacks to form part of this talk.

But I claim all credit for errors, and stupid mistakes. These are mine, all mine! You can’t have them.
Agenda

1. SYCL Past Present future
2. SYCL 2020
3. SYCL ecosystem
Khronos Active Initiatives and where does SYCL fit

3D Graphics
Desktop, Mobile, Web
Embedded and Safety Critical

3D Assets
Authoring and Delivery

Portable XR
Augmented and Virtual Reality

Parallel Computation
Vision, Inferencing, Machine Learning

Guidelines for creating APIs to streamline system safety certification
SYCL Single Source C++ Parallel Programming

C++ Kernel Fusion can give better performance on complex apps and libs than hand-coding.

Complex ML frameworks can be directly compiled and accelerated.

C++ templates and lambda functions separate host & accelerated device code.

SYCL is ideal for accelerating larger C++-based engines and applications with performance portability.
SYCL Implementations

SYCL, OpenCL and SPIR-V, as open industry standards, enable flexible integration and deployment of multiple acceleration technologies

Multiple Backends in Development
SYCL beginning to be supported on multiple low-level APIs in addition to OpenCL e.g. ROCm and CUDA
For more information: http://sycl.tech
SYCL Past up to 2017 SYCL 1.2.1

C++11  C++14  C++17

Work with industry to bring Heterogeneous compute to standard ISO C++

OpenCL ‘Next’ Flexible and efficient deployment of parallel computation across diverse processor architectures

2011
OpenCL 1.2
OpenCL C Kernel Language

2015
OpenCL 2.1
SPIR-V in Core

2017
OpenCL 2.2
C++ Kernel Language
SYCL Present and Future Roadmap (May Change)

<table>
<thead>
<tr>
<th>Year</th>
<th>SYCL</th>
<th>C++</th>
<th>Language</th>
<th>Other Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>SYCL 1.2</td>
<td>C++11</td>
<td>Single source programming</td>
<td>OpenCL 1.2</td>
</tr>
<tr>
<td>2015</td>
<td>SYCL 1.2.1</td>
<td>C++11</td>
<td>Single source programming</td>
<td>OpenCL 2.1</td>
</tr>
<tr>
<td>2017</td>
<td>SYCL 2020</td>
<td>C++17</td>
<td>Single source programming</td>
<td>OpenCL 2.2</td>
</tr>
<tr>
<td>2020</td>
<td>SYCL 2021</td>
<td>C++20</td>
<td>Single source programming</td>
<td>Many backend options</td>
</tr>
<tr>
<td>2021</td>
<td>SYCL 2021</td>
<td>C++20</td>
<td>Single source programming</td>
<td>Many backend options</td>
</tr>
</tbody>
</table>

- OpenCL 1.2
- OpenCL 2.1 SPIR-V in Core
- OpenCL 2.2
- OpenCL 3.0

© Copyright Khronos™ Group 2018 - Page 8
SYCL community is vibrant

SYCL F2F meetings attendance

2X growth

SYCL-1.2.1
Contributors (Thank you, SYCL WG!)

- Ronan Keryell (Xilinx)
- Brian Sumner (AMD)
- Alexey Bader (Intel)
- James Brodman (Intel)
- Mike Kinsner (Intel)
- John Pennycook (Intel)
- Roland Schulz (Intel)
- Ilya Burylov (Intel)
- Jeff Hammond (Intel)
- Ben Ashbaugh (Intel)
- Andrew Gozillon (Xilinx)
- Aksel Alpay (Self)
- Daniel Berenyi (Self)
- Máté Nagy-Egri (StreamHPC)
- Gordon Brown (CP)
- Victor Lomüller (CP)
- Peter Žužek (CP)
- Morris Hafner (CP)
- Andrew Richards (CP)
- Michael Wong (CP)
- Ruymán Reyes (CP)
- Steffen Larsen (CP)
- Stuart Adams (CP)
- Alex Johnston (CP)
- Tomas Matheson (CP)
- Victor Lezaud (Xilinx)
- Sébastien Le Duc (Kalray)
- Kévin Petit (ARM)
- Anastasia Stulova (ARM)
- Nevin Liber (ANL)
- Brian Homerding (ANL)
- Hal Finkel (ANL)
- Neil Trevett (Khronos)
- Tom Deakin (U of Bristol)
- Ruihao Zhang (Qualcomm)
- Biagio Cosenza (Salerno)
- And many others
+ YOU on GitHub!
Commits by year/month (thanks Ronan Keryell, Xilinx)
Agenda

1. SYCL Past Present future
2. SYCL 2020
3. SYCL ecosystem
Extending SYCL to the future

This is a preview and describes work in progress. The content might change. Some of these features have not been merged.
GitHub/Gitlab open-source collaboration

Your great contributions

Public GitHub

SYCL-1.2.1/squashed-rev-5
Squashed history

SYCL-1.2.1/master

Pull

Cherry-pick close-source developments

Private GitLab

SYCL-1.2.1/pub-rev-5

pub/SYCL-1.2.1/master

SYCL-1.2.1/master

SYCL-1.2.1/final-rev5

Our IP-secret sauce, to be open-sourced after ratification
SYCL Evolution

SYCL 2020 potential Features
- Generalization (a.k.a the Backend Model) presented by Gordon Brown
- Unified Shared Memory (USM) presented by James Brodman
- Improvement to Program class Modules presented by Gordon Brown
- Host Task with Interop presented by Gordon Brown
- In order queues, presented by James Brodman

SYCL 2020 compared with SYCL 1.2.1
- Easier to integrate with C++17 (CTAD, Deduction Guides,...)
- Less verbose, smaller code size, simplify patterns
- Backend independent
- Multiple object archives aka modules simplify interoperability
- Ease porting C++ applications to SYCL
- Enable capabilities to improve programmability
- Backwards compatible but Minor API break based on user feedback

SYCL Evolution

2017
SYCL 1.2.1

Improving Software Ecosystem
- Tool, libraries, GitHub

Expanding Implementation
- DPC++
- ComputeCPP
- triSYCL
- hipSYCL

Regular Maintenance Updates
- Spec clarifications, formatting and bug fixes
  https://www.khronos.org.registry/SYCL/

Repeat The Cycle every
1.5-3 years

Target 2020
Provisional Q3 then Final Q4

Selected Extension Pipeline aiming for
SYCL 2020 Provisional Q3
- Reduction
- Subgroups
- Accessor simplification
- Atomic rework
- Extension mechanism
- Address spaces
- Vector rework
- Specialization Constants

Converge SYCL with ISO C++ and continue to support OpenCL on more devices

CPU
GPU
FPGA
AI Processors
Custom Processors

SYCL 2020 Roadmap (WIP, MAY CHANGE)
Agenda

1. SYCL Past Present future
2. SYCL 2020
3. SYCL ecosystem
SYCL Ecosystem, Research and Benchmarks

Implementations

oneAPI

DATA PARALLEL C++

Research

Celerity

ECP

BabelStream

SYCL-Blas

SYCL-BLAS

Linear Algebra Libraries

SYCL-ML

Machine Learning Libraries and Parallel Acceleration Frameworks

SYCL-DNN

Eigen

oneMKL

SYCL Parallel STL

Implementations

RSBench

Implementations

Background Parallel Research Kernels

Create tests to study behavior of parallel systems
- Cover a broad range of patterns found in real parallel applications
- Provide a paper-and-pencil specification and a reference implementation
- Keep kernels simple functionally
- Easy to understand in different domains
- Easy to understand in different domains
- Optimized for a single system, an inherent performance benchmark block
- Parameters in kernels (number of iterations, error, etc.)
- Make sure each kernel does actual work
- Include automatic verification test (automatic solution)
- Ensure enough executable concurrency (can be load balanced)
- Make sure each kernel does actual work

Implementations

Active Working Group Members

oneAPI

ComputeCpp

hipSYCL

Argonne National Laboratory

Arm

SYCL

Intel

Xilinx

codeplay

Implementations

University of Bristol
## SYCL, Aurora and Exascale computing

<table>
<thead>
<tr>
<th>Program</th>
<th>Laboratory</th>
<th>Timeline/Projected timeline</th>
<th>System Name/Prime Contractor</th>
<th>System Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORAL-1</td>
<td>ANL</td>
<td>System delivered in late 2021 and accepted in 2022</td>
<td>Aurora/Intel</td>
<td>Cray Shasta with Intel Xe GPUs</td>
</tr>
<tr>
<td>CORAL-2</td>
<td>ORNL</td>
<td>System delivered in late 2021 and accepted in 2022</td>
<td>Frontier/Cray</td>
<td>Cray Shasta with AMD future Epyc CPUs and future Radeon GPUs</td>
</tr>
<tr>
<td>CORAL-2</td>
<td>LLNL</td>
<td>System delivered in late 2022 and accepted in late 2023</td>
<td>El Capitan/Cray</td>
<td>Cray Shasta with CPUs and GPUs</td>
</tr>
</tbody>
</table>

SYCL can run on AMD ROCM
SYCL future and C++ roadmap

Integrated more tightly with ISO C++, members attending both

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1980</td>
<td></td>
</tr>
<tr>
<td>1981</td>
<td></td>
</tr>
<tr>
<td>1982</td>
<td></td>
</tr>
<tr>
<td>1983</td>
<td></td>
</tr>
<tr>
<td>1984</td>
<td></td>
</tr>
<tr>
<td>1985</td>
<td></td>
</tr>
<tr>
<td>1986</td>
<td></td>
</tr>
<tr>
<td>1987</td>
<td></td>
</tr>
<tr>
<td>1988</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td></td>
</tr>
<tr>
<td>1991</td>
<td></td>
</tr>
<tr>
<td>1992</td>
<td></td>
</tr>
<tr>
<td>1993</td>
<td></td>
</tr>
<tr>
<td>1994</td>
<td></td>
</tr>
<tr>
<td>1995</td>
<td></td>
</tr>
<tr>
<td>1996</td>
<td></td>
</tr>
<tr>
<td>1997</td>
<td></td>
</tr>
<tr>
<td>1998</td>
<td></td>
</tr>
<tr>
<td>1999</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
</tr>
</tbody>
</table>

### IS: trunk

- C++0x/11
- Library TR1
- Decimal TR (not merged)
- Math Special Functions IS
- Library TR2 (deferred to post-C++17, then replaced by File System TS)

### TSes: feature branches for separate release & then merge

<table>
<thead>
<tr>
<th>Year</th>
<th>Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>Aug</td>
</tr>
<tr>
<td></td>
<td>Feb</td>
</tr>
<tr>
<td></td>
<td>Oct</td>
</tr>
<tr>
<td>2012</td>
<td>Apr</td>
</tr>
<tr>
<td></td>
<td>Sep</td>
</tr>
<tr>
<td></td>
<td>Feb</td>
</tr>
<tr>
<td>2013</td>
<td>Jun</td>
</tr>
<tr>
<td></td>
<td>Nov</td>
</tr>
<tr>
<td>2014</td>
<td>May</td>
</tr>
<tr>
<td></td>
<td>Oct</td>
</tr>
<tr>
<td>2015</td>
<td>Mar</td>
</tr>
<tr>
<td></td>
<td>Jun</td>
</tr>
<tr>
<td></td>
<td>Nov</td>
</tr>
<tr>
<td>2016</td>
<td>Mar</td>
</tr>
<tr>
<td></td>
<td>Jul</td>
</tr>
<tr>
<td></td>
<td>Nov</td>
</tr>
<tr>
<td>2017</td>
<td>Mar</td>
</tr>
<tr>
<td></td>
<td>Jun</td>
</tr>
<tr>
<td></td>
<td>Nov</td>
</tr>
<tr>
<td>2018</td>
<td>Feb</td>
</tr>
<tr>
<td></td>
<td>Jul</td>
</tr>
<tr>
<td></td>
<td>Nov</td>
</tr>
<tr>
<td>2019</td>
<td>Feb</td>
</tr>
</tbody>
</table>

### C++14
- File System
- Lib Fundamentals 1
- Parallelism 1
- Concepts
- Arrays (abandoned)

### C++17
- Networking
- Lib Fundamentals 2
- Parallelism 2
- Ranges
- Modules

### C++20
- Reflection
- Lib Fundamentals 3
- Concurrency 1
- Coroutines

TS bars start and end where work on detailed specification wording starts ("adopt initial working draft") and ends ("send to publication") Future starts/ends are shaded to indicate that dates, and TS branches are approximate and subject to change.
SYCL 2020 provisional is coming

- In a few months, SYCL 2020 provisional will be released
- We need your feedback asap
  - https://app.slack.com/client/TDMDFS87M/CE9UX4CHG
  - https://community.khronos.org/c/sycl/
  - https://sycl.tech/
- What features are you looking for that is not in SYCL 2020?
- What feature would you like to aim for in future SYCL?
- How do you join SYCL?
Engaging with the Khronos SYCL Ecosystem

**Contribute to SYCL open source specs, CTS, tools and ecosystem**

**SYCL Working Groups**

**SYCL Advisory Panels**

**Khronos SYCL Forums, Slack Channels, stackoverflow, reddit, and SYCL.tech**

Open to all!
- [community.khronos.org](https://community.khronos.org)
- [app.slack.com/client/TDMDFS87M/CE9UX4CHG](https://app.slack.com/client/TDMDFS87M/CE9UX4CHG)
- [community.khronos.org/c/sycl](https://community.khronos.org/c/sycl)
- [stackoverflow.com/questions/tagged/sycl](https://stackoverflow.com/questions/tagged/sycl)
- [www.reddit.com/r/sycl](https://www.reddit.com/r/sycl)
- [github.com/codeplaysoftware/syclacademy](https://github.com/codeplaysoftware/syclacademy)
- [https://sycl.tech/](https://sycl.tech/)

Any member or non-member can propose a new SYCL feature or fix

Spec fixes and suggestions made under the Khronos IP Framework. Open source contributions under repo's CLA - typically Apache 2.0
- [github.com/KhronosGroup](https://github.com/KhronosGroup)
- [github.com/KhronosGroup/SYCL-CTS](https://github.com/KhronosGroup/SYCL-CTS)
- [github.com/KhronosGroup/SYCL-Docs](https://github.com/KhronosGroup/SYCL-Docs)
- [github.com/KhronosGroup/SYCL-Shared](https://github.com/KhronosGroup/SYCL-Shared)
- [github.com/KhronosGroup/SYCL-Registry](https://github.com/KhronosGroup/SyclParallelSTL)

Invited Advisors under the Khronos NDA and IP Framework can comment and contribute to requirements and draft specifications
- [https://www.khronos.org/advisors/](https://www.khronos.org/advisors/)

Khronos members under Khronos NDA and IP Framework participate and vote in working group meetings. Starts at $3.5K/yr.
- [https://www.khronos.org/members/](https://www.khronos.org/members/)
- [https://www.khronos.org/registry/SYCL/](https://www.khronos.org/registry/SYCL/)
Thank You!

- Khronos SYCL is creating cutting-edge royalty-free open standard
  - For C++ Heterogeneous compute, vision, inferencing acceleration

- Information on Khronos SYCL Standards: https://www.khronos.org/sycl/
- Any entity/individual is welcome to join Khronos SYCL: https://www.khronos.org/members/
- Join the SYCLCon Tutorial Monday and Wednesday Live panel: Wednesday Apr 29 15:00-18:00 GMT
  - Have your questions answered live by a group of SYCL experts
- Michael Wong: michael@codeplay.com | wongmichael.com/about

Benefits of Khronos membership:

- Gather industry requirements for future open standards
- Draft Specifications Confidential to Khronos members
- Publicly Release Specifications and Conformance Tests
- Gain early insights into industry trends and directions
- Influence the design and direction of key open standards that will drive your business
- Accelerate your time-to-market with early access to specification drafts
- Network with domain experts from diverse companies in your industry
- State-of-the-art IP Framework protects your Intellectual Property
- Enhance your company reputation as an industry leader through Khronos participation