The 11th International workshop on OpenCL and SYCL

IWOCL & SYCLcon 2023

What's New in SYCL SC™ (SYCL for Safety Critical Systems)

Erik Tomusk, Codeplay Software

Verena Beckham, Codeplay Software

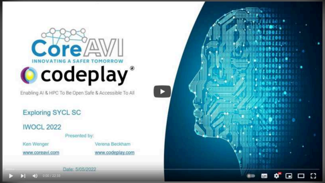
April 18-20, 2023 | University of Cambridge, UK iwocl.org

Introduction

Notices & Disclaimers
Codeplay Software Ltd., Codeplay, Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Introduction

- Follow-up to last year's presentation
  - https://www.youtube.com/watch?v=tD46tGOglUg



CoreAVI
INNOVATING A SAFER TOMORROW

codeplay®
Enabling AI & HPC To Be Open Safe & Accessible To All

Exploring SYCL SC

IWOCL 2022

Presented by:

Ken Wenger
www.coreavi.com

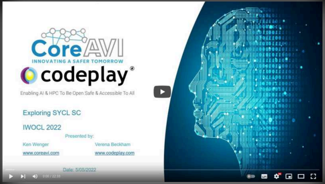
Verena Beckham
www.codeplay.com

Date: 5/05/2022

Notices & Disclaimers
Codeplay Software Ltd., Codeplay, Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Introduction

- Follow-up to last year's presentation
  - https://www.youtube.com/watch?v=tD46tGOglUg
- Short version:



CoreAVI

INNOVATING A SAFER TOMORROW

codeplay®

Enabling AI & HPC To Be Open Safe & Accessible To All

Exploring SYCL SC

IWOCL 2022

Presented by:

Ken Wenger www.coreavi.com

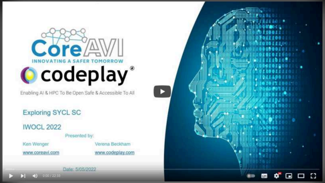
Verena Beckham www.codeplay.com

Date: 5/05/2022

Notices & Disclaimers
Codeplay Software Ltd., Codeplay, Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Introduction

- Follow-up to last year’s presentation
  - https://www.youtube.com/watch?v=tD46tGOglUg
- Short version:
  - Had a Khronos exploratory forum (EF) to investigate demand for and scope of SYCL SC



CoreAVI
INNOVATING A SAFER TOMORROW

codeplay®
Enabling AI & HPC To Be Open Safe & Accessible To All

Exploring SYCL SC

IWOCL 2022

Presented by:

Ken Wenger
www.coreavi.com

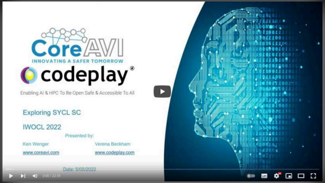
Verena Beckham
www.codeplay.com

Date: 5/05/2022

Notices & Disclaimers
Codeplay Software Ltd., Codeplay, Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Introduction

- Follow-up to last year's presentation
  - https://www.youtube.com/watch?v=tD46tGOglUg
- Short version:
  - Had a Khronos exploratory forum (EF) to investigate demand for and scope of SYCL SC
  - The EF produced a statement of work (SoW)



CoreAVI
INNOVATING A SAFER TOMORROW

codeplay®
Enabling AI & HPC To Be Open Safe & Accessible To All

Exploring SYCL SC

IWOCL 2022

Presented by:

Ken Wenger
www.coreavi.com

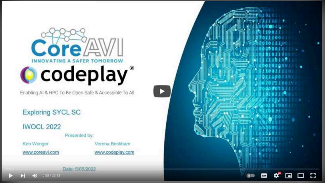
Verena Beckham
www.codeplay.com

Date: 5/05/2022

Notices & Disclaimers
Codeplay Software Ltd., Codeplay, Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Introduction

- Follow-up to last year's presentation
  - https://www.youtube.com/watch?v=tD46tGOglUg
- Short version:
  - Had a Khronos exploratory forum (EF) to investigate demand for and scope of SYCL SC
  - The EF produced a statement of work (SoW)
  - Approved March 2023



CoreAVI
INNOVATING A SAFER TOMORROW
codeplay®
Enabling AI & HPC To Be Open Safe & Accessible To All
0:00 / 22:33

Exploring SYCL SC

IWOCL 2022

Presented by:

Ken Wenger
www.coreavi.com

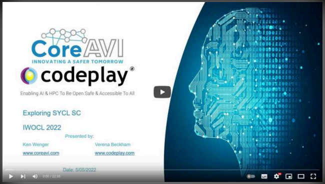
Verena Beckham
www.codeplay.com

Date: 5/05/2022

Notices & Disclaimers
Codeplay Software Ltd., Codeplay, Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Introduction

- Follow-up to last year's presentation
  - https://www.youtube.com/watch?v=tD46tGOglUg
- Short version:
  - Had a Khronos exploratory forum (EF) to investigate demand for and scope of SYCL SC
  - The EF produced a statement of work (SoW)
  - Approved March 2023
  - EF finished; now there is a working group (WG)



CoreAVI
INNOVATING A SAFER TOMORROW

codeplay®
Enabling AI & HPC To Be Open Safe & Accessible To All

Exploring SYCL SC

IWOCL 2022

Presented by:

Ken Wenger
www.coreavi.com

Verena Beckham
www.codeplay.com

Date: 5/05/2022

Notices & Disclaimers
Codeplay Software Ltd., Codeplay, Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Industry Support

The deployment of autonomous systems is increasingly dependent on open standards and a robust software and hardware ecosystem that places safety as its top priority.

Tom Conway
(Sr. Director, Product Management, Automotive)

arm

Mercedes-Benz Research & Development North America is delighted to join the SYCL SC Working Group to strengthen industry adoption of safety-critical development standards, as ADAS/AD systems harness heterogeneous computing hardware.

Sundararajan Ramalingam
(VP Autonomous Driving)

Mercedes-Benz
Research & Development North America

SYCL Safety Critical extends open accelerator programming to applications where safety critical standards apply.

Joe Curly
(VP Software Products)

intel

AMD

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Qualcomm

intellias

CoreAVI
INNOVATING A SAFER TOMORROW

codeplay

https://www.khronos.org/news/press/khronos-to-create-sycl-sc-open-standard-for-safety-critical-c-based-heterogeneous-compute

Outline

Outline

1. Background
2. Summary of SYCL SC EF activities
3. Technical lessons learned
   - SYCL SC Scope
   - SC dynamic memory
4. Conclusions

Outline

1. Background
2. Summary of SYCL SC EF activities
3. Technical lessons learned
    - SYCL SC Scope
    - SC dynamic memory
4. Conclusions

- Caveats:
    - Speaking as a Codeplay employee
    - Not representing Khronos
    - Not representing SYCL SC EF or SYCL SC WG
    - SYCL SC EF was under NDA

Outline

1. **Background**
2. Summary of SYCL SC EF activities
3. Technical lessons learned
   - SYCL SC Scope
   - SC dynamic memory
4. Conclusions

Background: “Safety Critical”

Background: “Safety Critical”

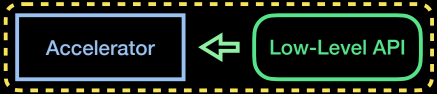
- For software, a safety-critical domain is one where a software malfunction could cause serious damage
  - To people
  - To property
  - To the environment

Background: “Safety Critical”

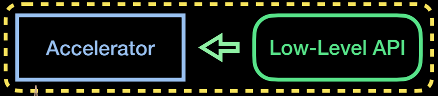
- For software, a safety-critical domain is one where a software malfunction could cause serious damage
  - To people
  - To property
  - To the environment
- Domains include
  - Automotive
  - Aerospace
  - Medical
  - Nuclear
  - Rail
  - …

Background: Why SYCL?

Background: Why SYCL?

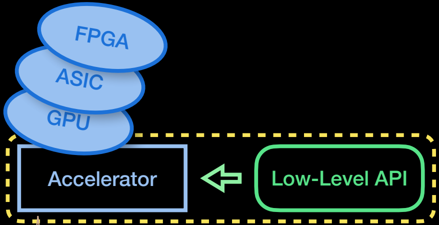


Background: Why SYCL?



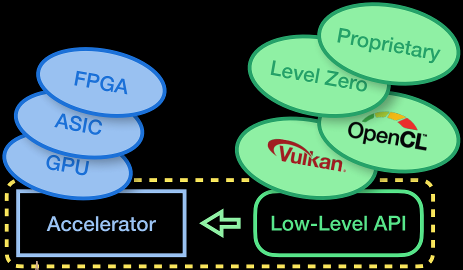
Much faster than CPU for some tasks

Background: Why SYCL?



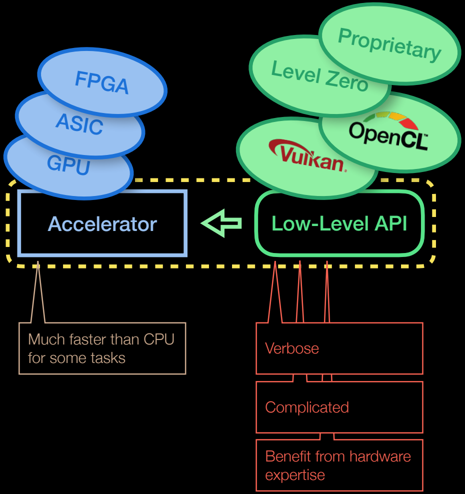
Much faster than CPU for some tasks

Background: Why SYCL?



Much faster than CPU for some tasks

Background: Why SYCL?



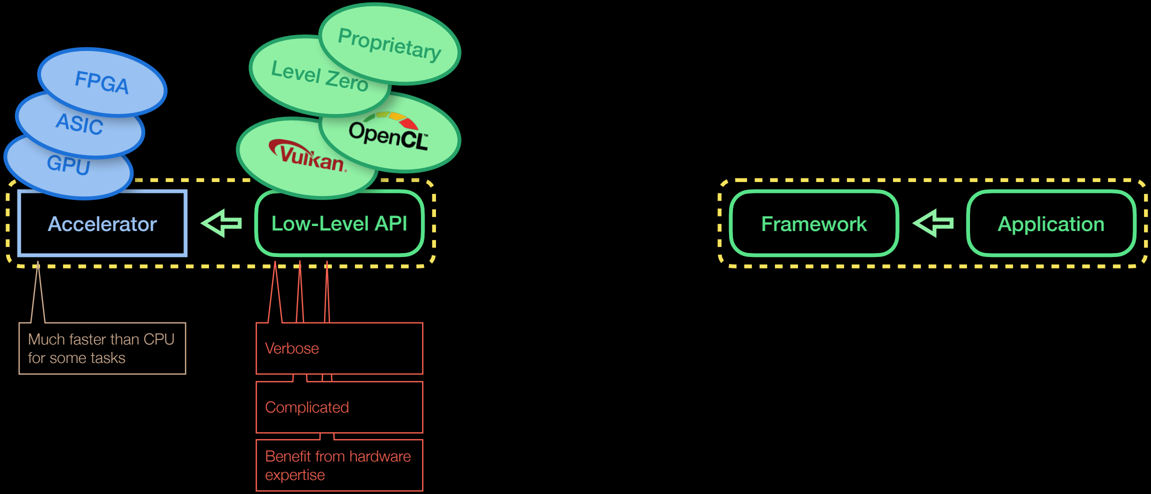
Much faster than CPU for some tasks

Verbose

Complicated

Benefit from hardware expertise

Background: Why SYCL?



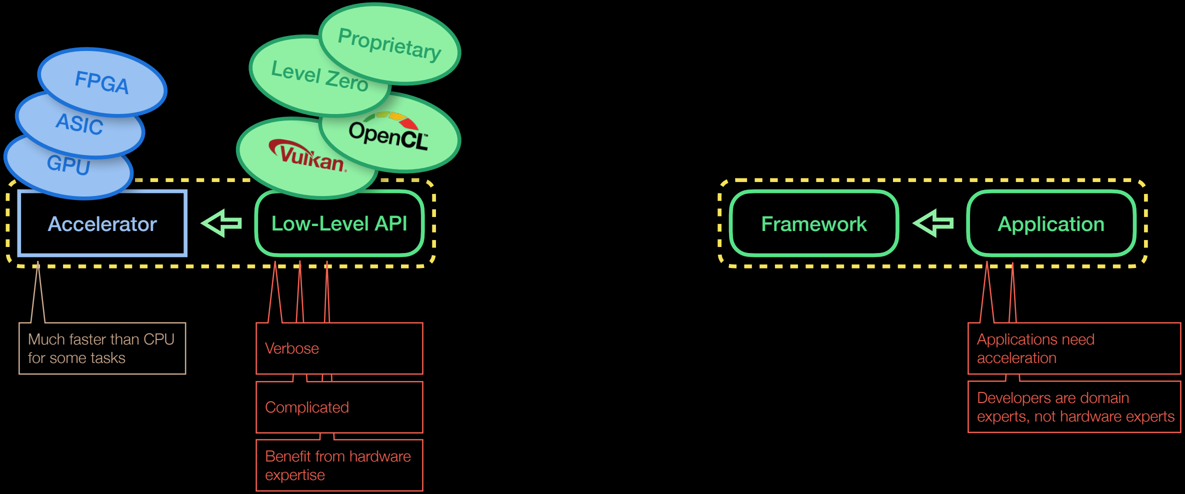
Much faster than CPU for some tasks

Verbose

Complicated

Benefit from hardware expertise

Background: Why SYCL?



Much faster than CPU for some tasks

Verbose

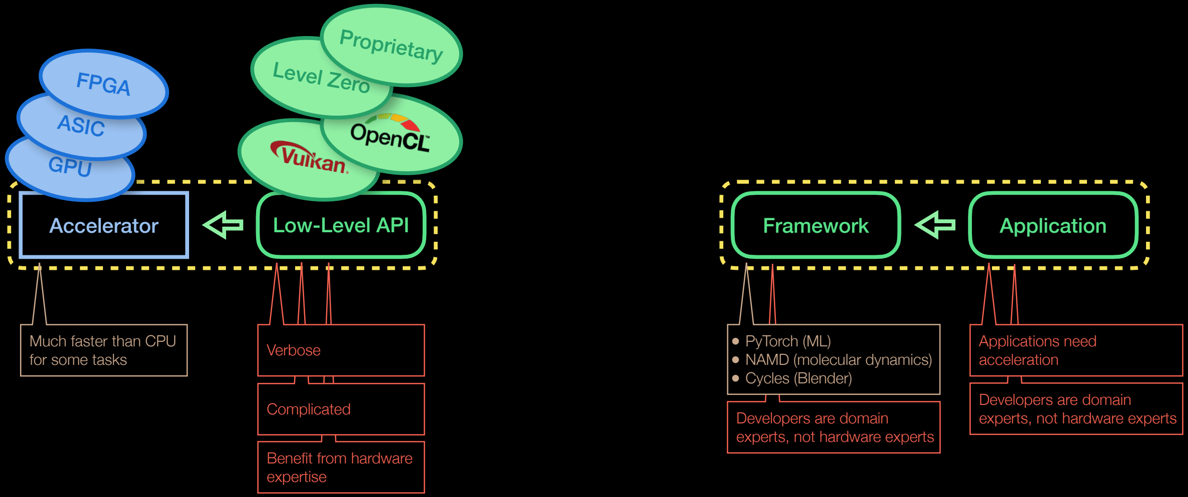
Complicated

Benefit from hardware expertise

Applications need acceleration

Developers are domain experts, not hardware experts

Background: Why SYCL?



Accelerator ⇦ Low-Level API

Framework ⇦ Application

Much faster than CPU for some tasks

Verbose

Complicated

Benefit from hardware expertise

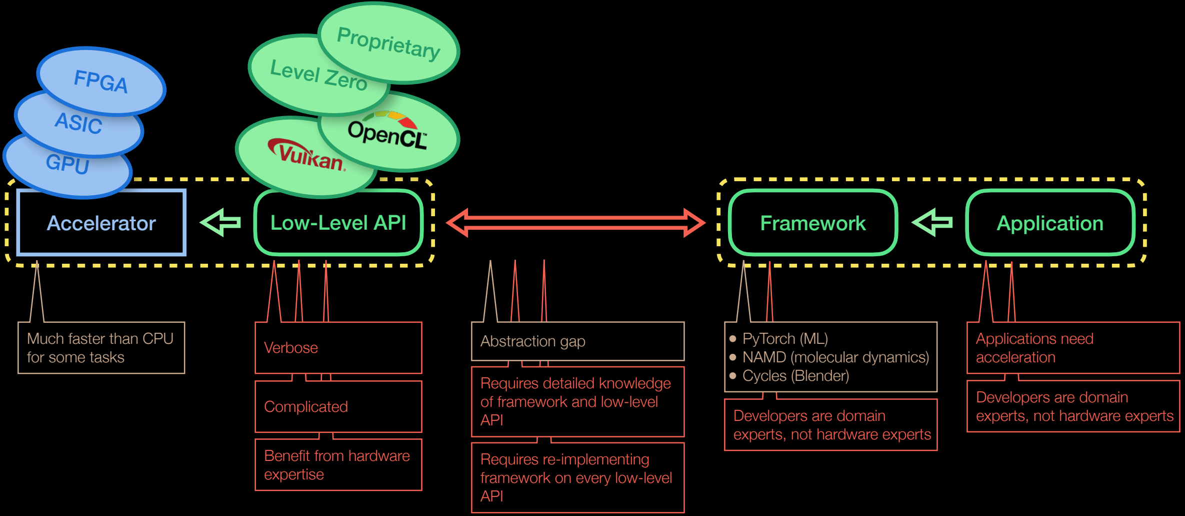
- PyTorch (ML)
- NAMD (molecular dynamics)
- Cycles (Blender)

Developers are domain experts, not hardware experts

Applications need acceleration

Developers are domain experts, not hardware experts

Background: Why SYCL?



Much faster than CPU for some tasks

Verbose

Complicated

Benefit from hardware expertise

Abstraction gap

Requires detailed knowledge of framework and low-level API

Requires re-implementing framework on every low-level API

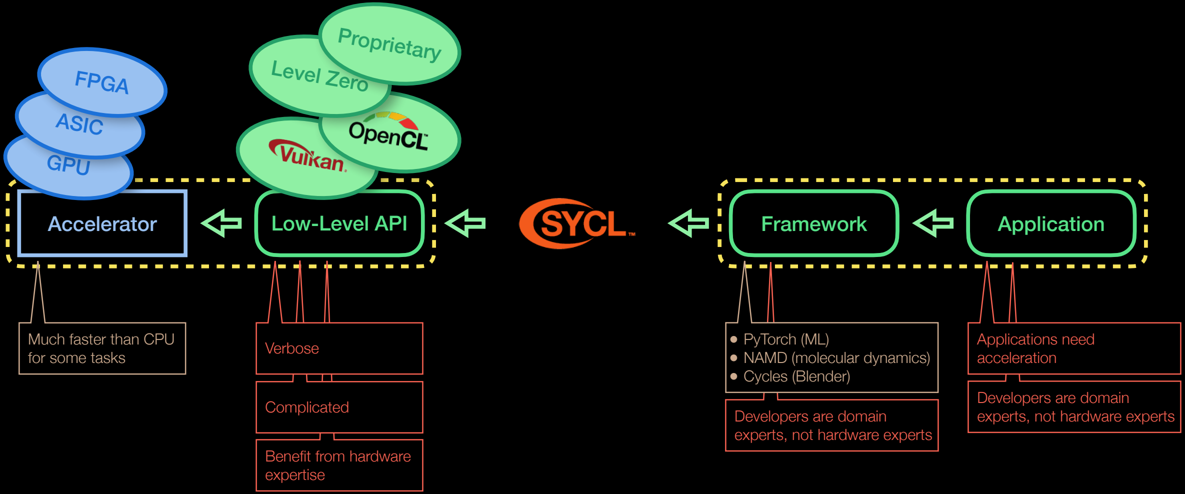
- PyTorch (ML)
- NAMD (molecular dynamics)
- Cycles (Blender)

Developers are domain experts, not hardware experts

Applications need acceleration

Developers are domain experts, not hardware experts

Background: Why SYCL?



Much faster than CPU for some tasks

Verbose

Complicated

Benefit from hardware expertise

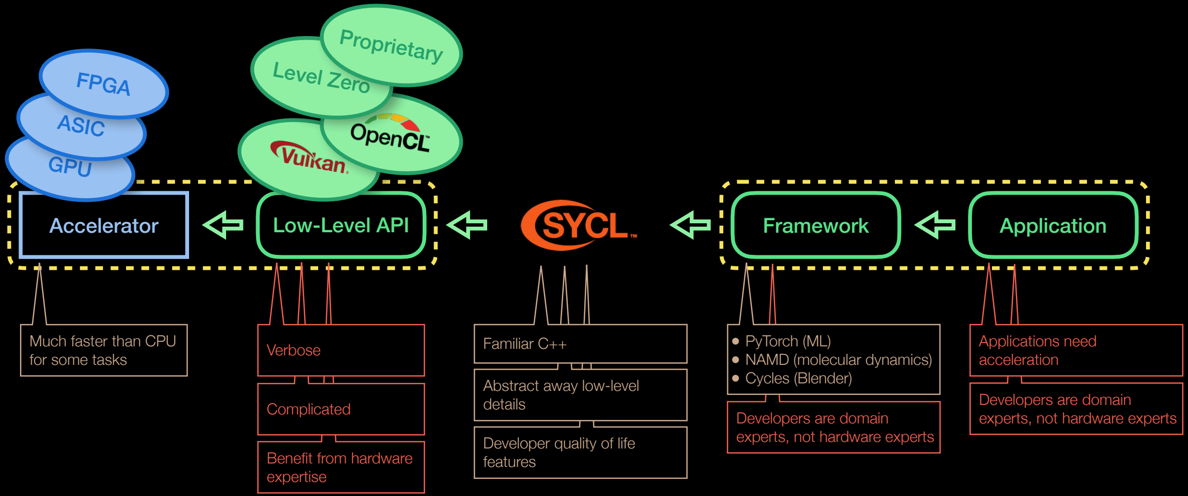
- PyTorch (ML)
- NAMD (molecular dynamics)
- Cycles (Blender)

Developers are domain experts, not hardware experts

Applications need acceleration

Developers are domain experts, not hardware experts

Background: Why SYCL?



Much faster than CPU for some tasks

Verbose

Complicated

Benefit from hardware expertise

Familiar C++

Abstract away low-level details

Developer quality of life features

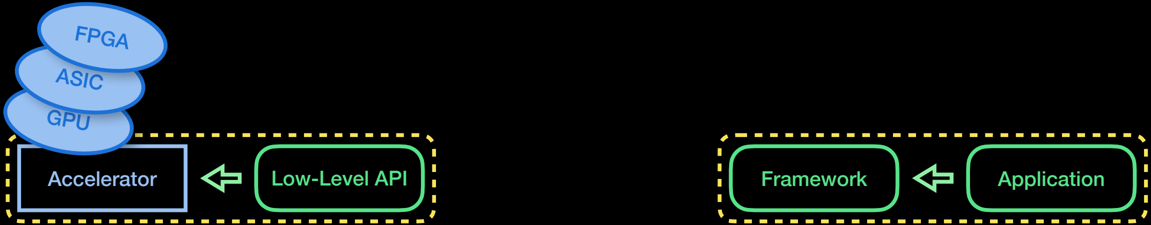
- PyTorch (ML)
- NAMD (molecular dynamics)
- Cycles (Blender)

Developers are domain experts, not hardware experts

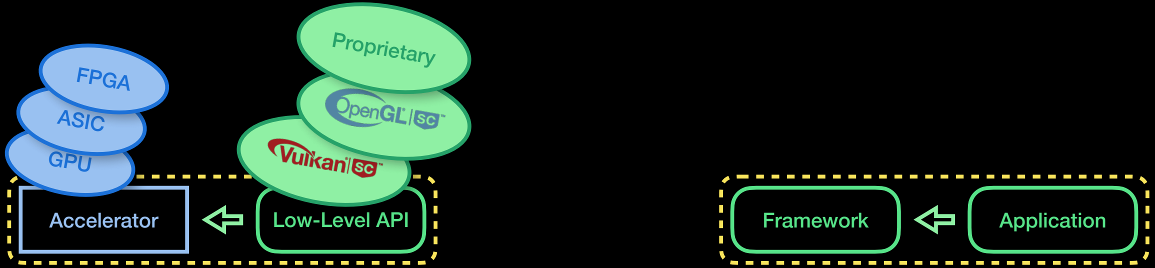
Applications need acceleration

Developers are domain experts, not hardware experts

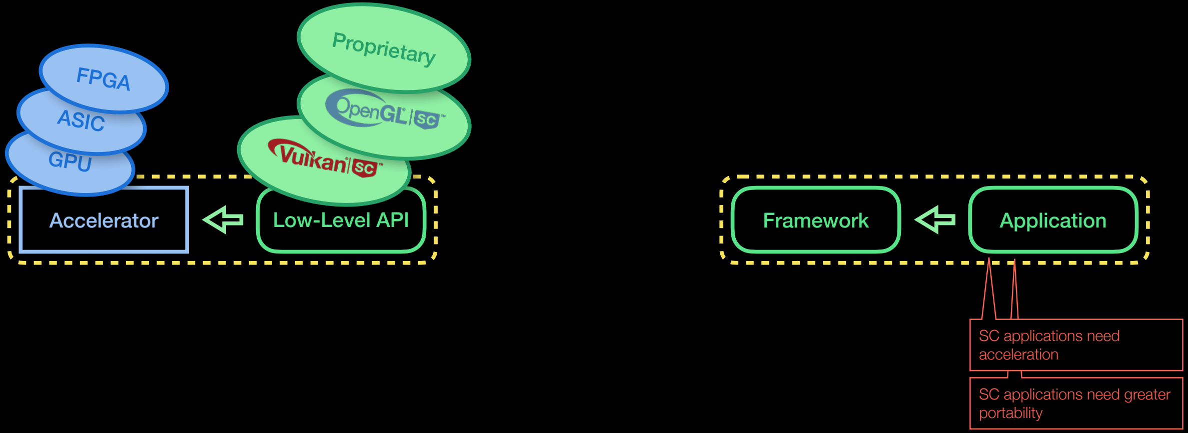
Background: Why SYCL SC?



Background: Why SYCL SC?



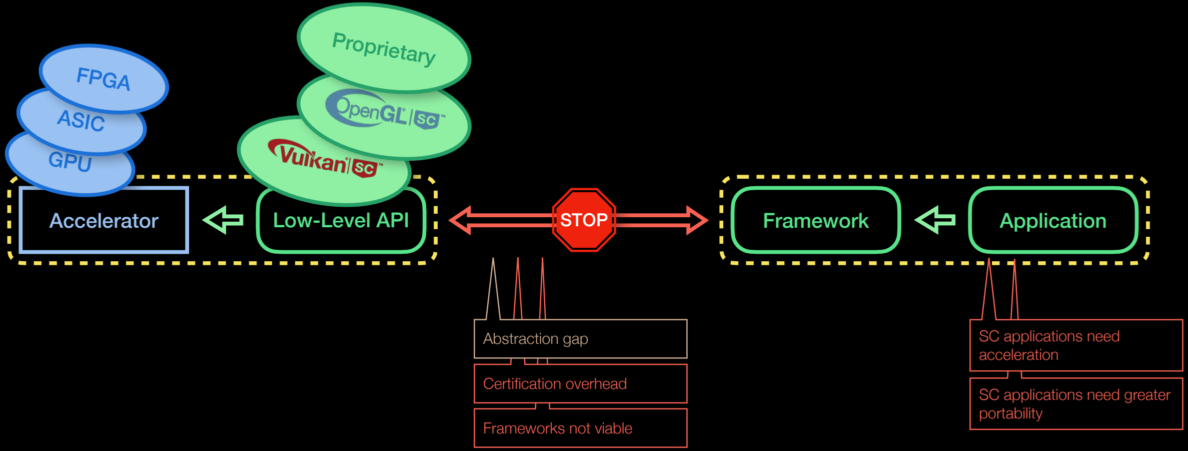
Background: Why SYCL SC?



SC applications need acceleration

SC applications need greater portability

Background: Why SYCL SC?



Abstraction gap

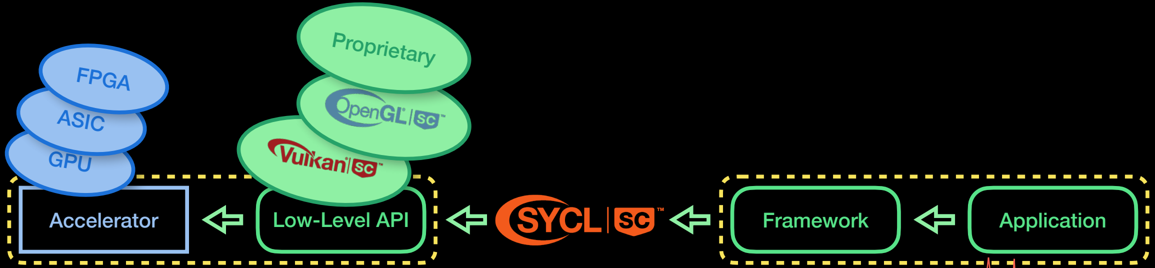
Certification overhead

Frameworks not viable

SC applications need acceleration

SC applications need greater portability

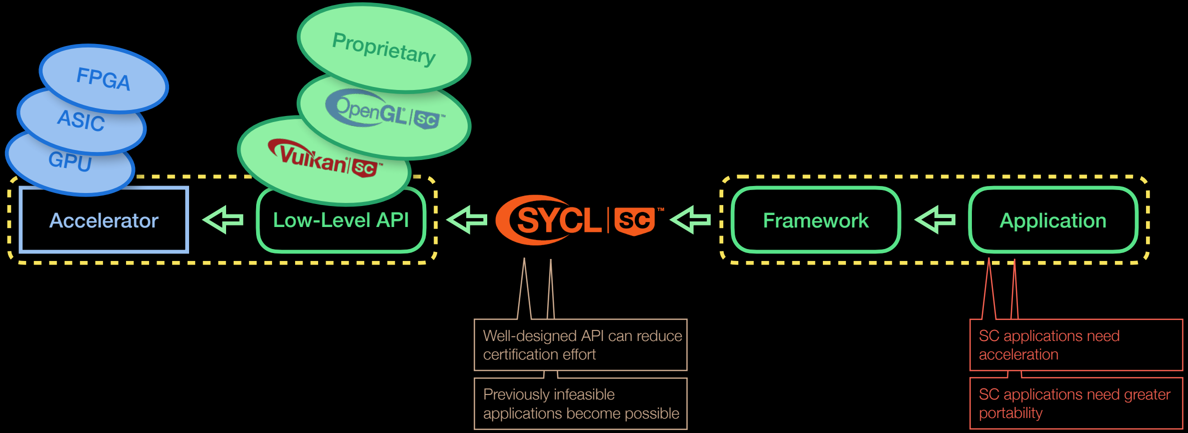
Background: Why SYCL SC?



SC applications need acceleration

SC applications need greater portability

Background: Why SYCL SC?



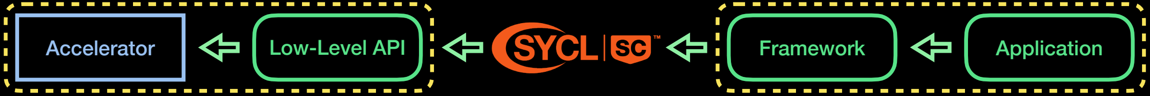
Well-designed API can reduce certification effort

Previously infeasible applications become possible

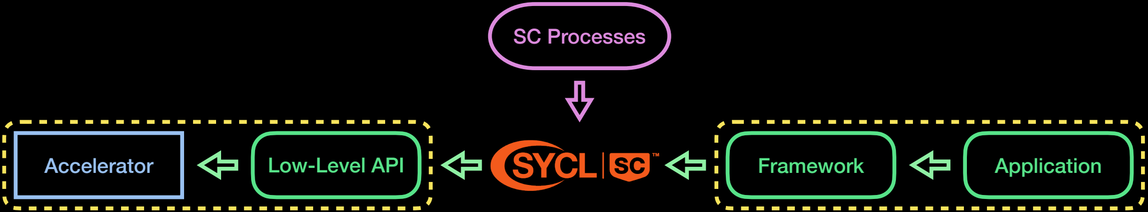
SC applications need acceleration

SC applications need greater portability

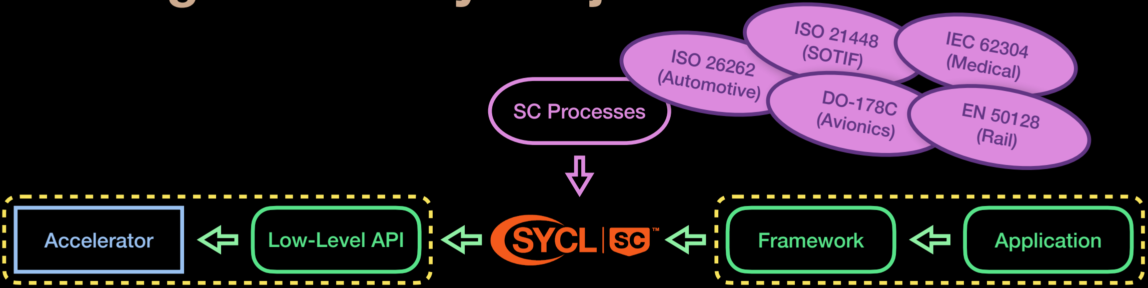
Background: Why not just SYCL?



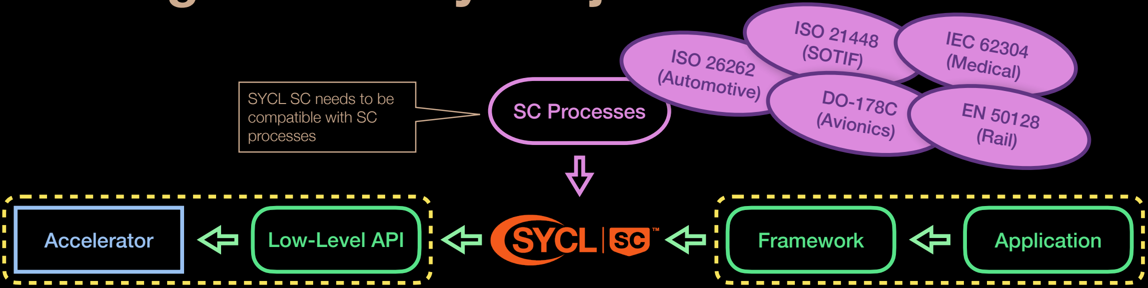
Background: Why not just SYCL?



Background: Why not just SYCL?



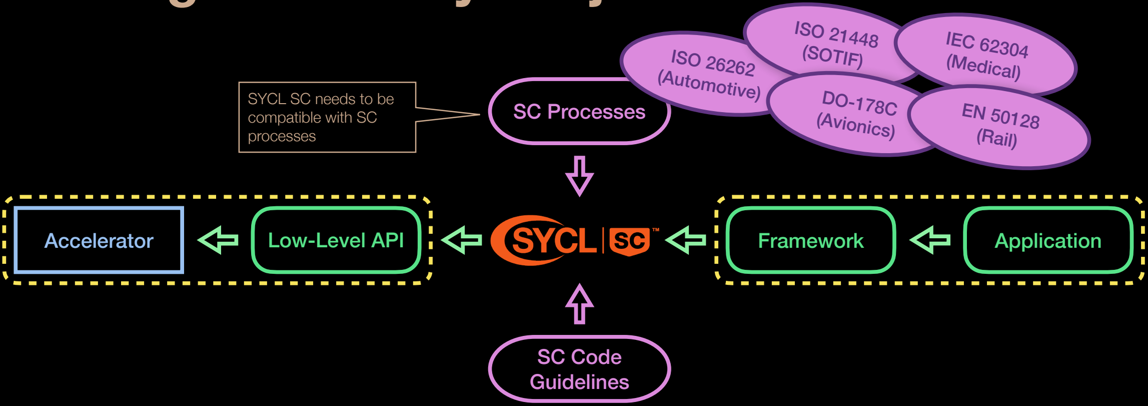
Background: Why not just SYCL?



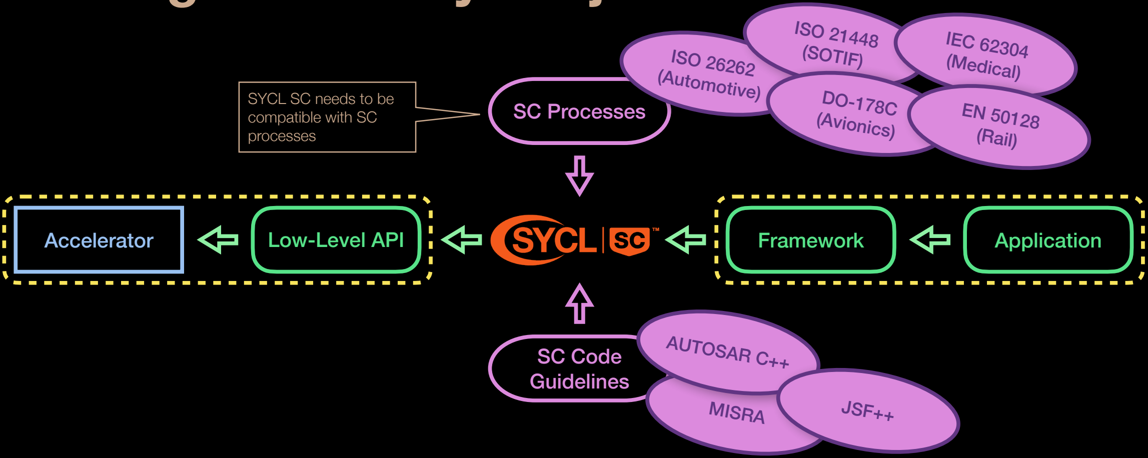
SYCL SC needs to be compatible with SC processes

Background: Why not just SYCL?

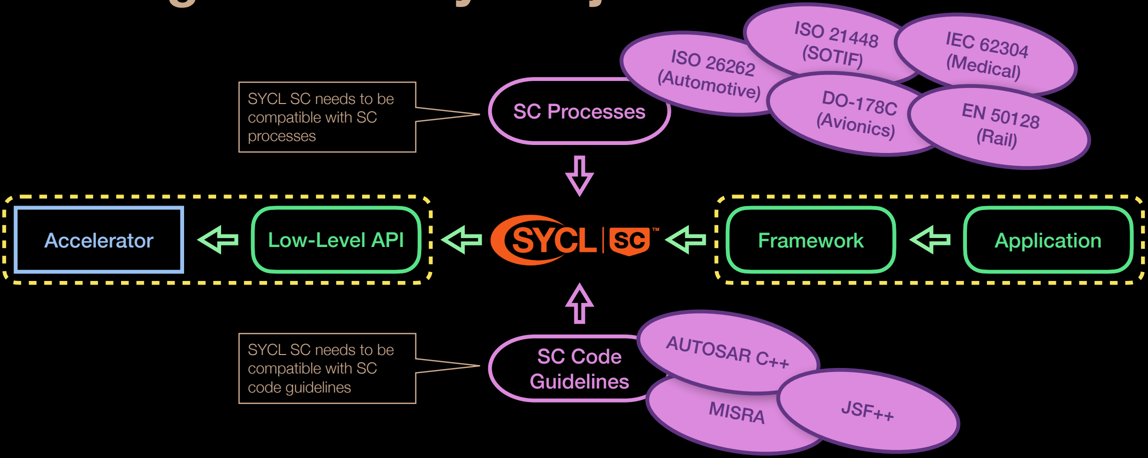
SYCL SC needs to be compatible with SC processes



Background: Why not just SYCL?



Background: Why not just SYCL?



SYCL SC needs to be compatible with SC processes

SC Processes

ISO 26262 (Automotive)

ISO 21448 (SOTIF)

IEC 62304 (Medical)

DO-178C (Avionics)

EN 50128 (Rail)

Accelerator

Low-Level API

SYCL SC™

Framework

Application

SYCL SC needs to be compatible with SC code guidelines

SC Code Guidelines

AUTOSAR C++

MISRA

JSF++

Background: Why not just SYCL?

Background: Why not just SYCL?

- SYCL SC philosophy

Background: Why not just SYCL?

- SYCL SC philosophy
  1. Enable predictable applications
     - Execution time, resource usage, results

Background: Why not just SYCL?

- SYCL SC philosophy
  1. Enable predictable applications
     - Execution time, resource usage, results
  2. Streamlined functionality to simplify runtime
     - Better compatibility with safety standards

Background: Why not just SYCL?

- SYCL SC philosophy
    1. Enable predictable applications
        - Execution time, resource usage, results
    2. Streamlined functionality to simplify runtime
        - Better compatibility with safety standards
    3. Specify comprehensive error handling, identify and remove ambiguities, and clarify undefined behavior
        - Better compatibility with safety-critical use cases

Background: Why not just SYCL?

- SYCL SC philosophy
  1. Enable predictable applications
     - Execution time, resource usage, results
  2. Streamlined functionality to simplify runtime
     - Better compatibility with safety standards
  3. Specify comprehensive error handling, identify and remove ambiguities, and clarify undefined behavior
     - Better compatibility with safety-critical use cases
- First-order priorities for SYCL SC, but lower priorities for SYCL

Background: What SYCL SC is not?

Background: What SYCL SC is not?

- SYCL SC intends to be *compatible* with SC processes and guidelines

Background: What SYCL SC is not?

- SYCL SC intends to be *compatible* with SC processes and guidelines
- The SYCL SC WG will *not …*

Background: What SYCL SC is not?

- SYCL SC intends to be *compatible* with SC processes and guidelines
- The SYCL SC WG will *not …*
  - Tell you how to implement a “safe” application
    - Using SYCL SC does not guarantee a safe application

Background: What SYCL SC is not?

- SYCL SC intends to be *compatible* with SC processes and guidelines
- The SYCL SC WG will *not …*
  - Tell you how to implement a “safe” application
    - Using SYCL SC does not guarantee a safe application
  - Tell you how to implement a “safe” SYCL runtime
    - Implementing the SYCL SC specification does not guarantee a safe runtime

Background: What SYCL SC is not?

- SYCL SC intends to be *compatible* with SC processes and guidelines
- The SYCL SC WG will *not …*
  - Tell you how to implement a "safe" application
    - Using SYCL SC does not guarantee a safe application
  - Tell you how to implement a "safe" SYCL runtime
    - Implementing the SYCL SC specification does not guarantee a safe runtime
  - Tell you how to apply any industry process or standard

Background: What SYCL SC is not?

- SYCL SC intends to be *compatible* with SC processes and guidelines
- The SYCL SC WG will *not …*
  - Tell you how to implement a “safe” application
    - Using SYCL SC does not guarantee a safe application
  - Tell you how to implement a “safe” SYCL runtime
    - Implementing the SYCL SC specification does not guarantee a safe runtime
  - Tell you how to apply any industry process or standard
  - Produce any certified code (tools or runtime)

Background: What SYCL SC is not?

- SYCL SC intends to be *compatible* with SC processes and guidelines
- The SYCL SC WG will *not …*
  - Tell you how to implement a “safe” application
    - Using SYCL SC does not guarantee a safe application
  - Tell you how to implement a “safe” SYCL runtime
    - Implementing the SYCL SC specification does not guarantee a safe runtime
  - Tell you how to apply any industry process or standard
  - Produce any certified code (tools or runtime)
    - Code snippets in SYCL SC specification will need to be *certifiable*

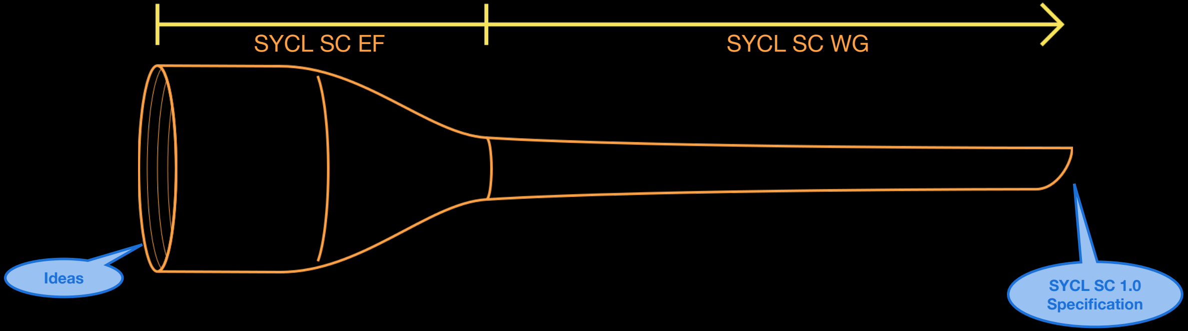
Background: What SYCL SC is not?

- SYCL SC intends to be *compatible* with SC processes and guidelines
- The SYCL SC WG will *not …*
  - Tell you how to implement a "safe" application
    - Using SYCL SC does not guarantee a safe application
  - Tell you how to implement a "safe" SYCL runtime
    - Implementing the SYCL SC specification does not guarantee a safe runtime
  - Tell you how to apply any industry process or standard
  - Produce any certified code (tools or runtime)
    - Code snippets in SYCL SC specification will need to be *certifiable*
- SYCL SC will be *compatible* with you doing the above, but cannot do it for you

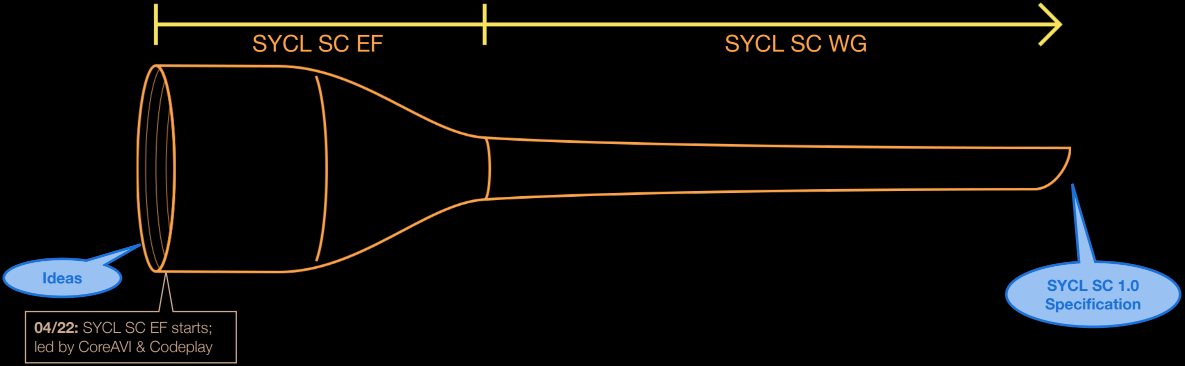
Outline

1. Background
2. **Summary of SYCL SC EF activities**
3. Technical lessons learned
   - SYCL SC Scope
   - SC dynamic memory
4. Conclusions

SYCL SC EF: Process

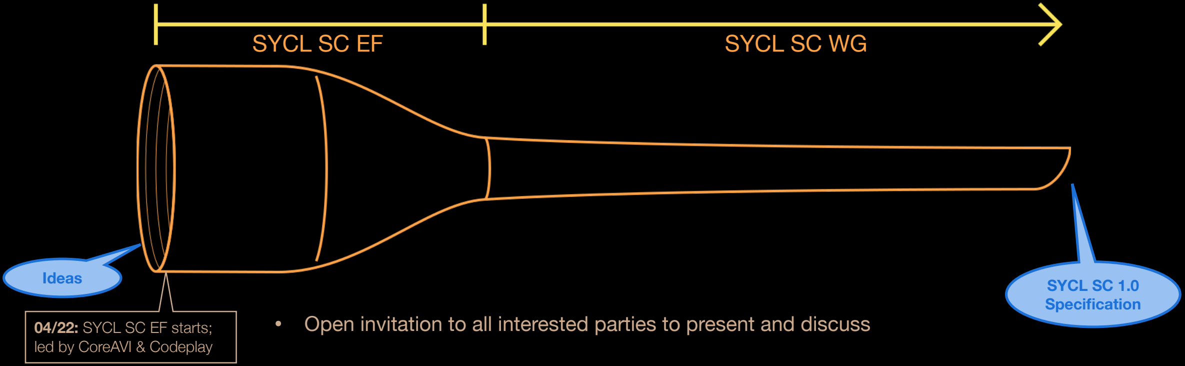


SYCL SC EF: Process



04/22: SYCL SC EF starts; led by CoreAVI & Codeplay

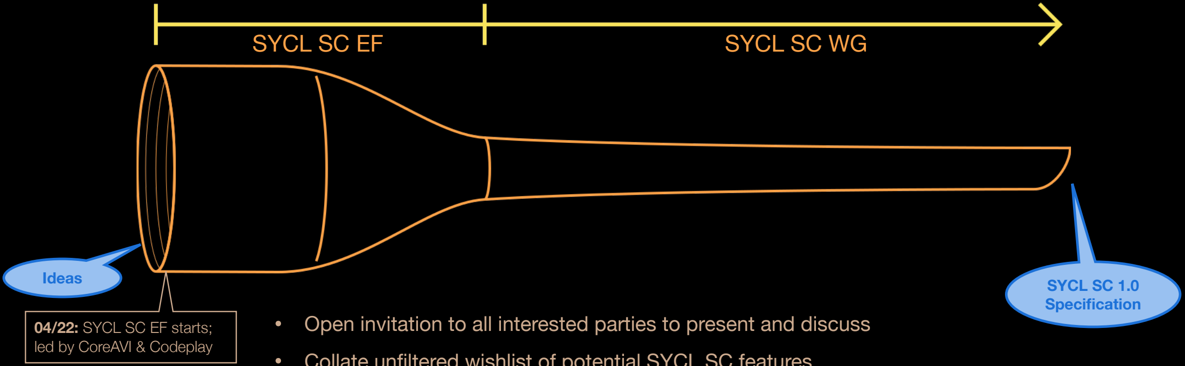
SYCL SC EF: Process



04/22: SYCL SC EF starts; led by CoreAVI & Codeplay

- Open invitation to all interested parties to present and discuss

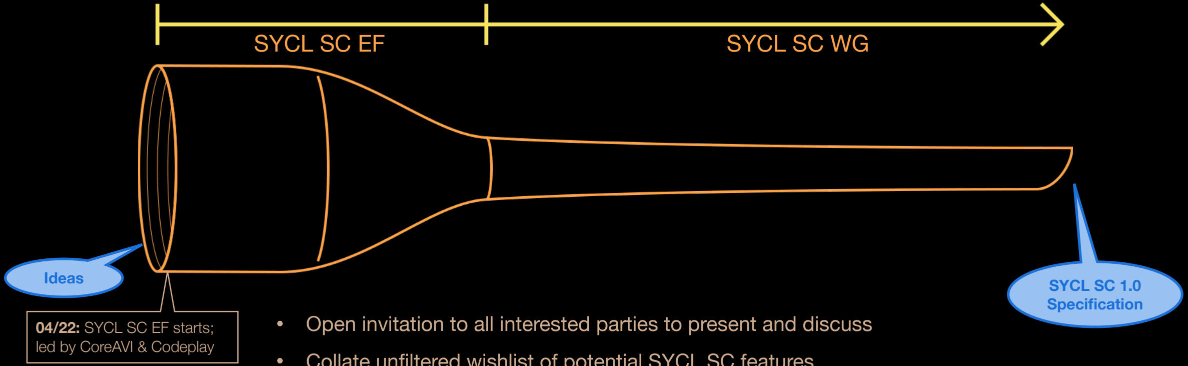
SYCL SC EF: Process



04/22: SYCL SC EF starts; led by CoreAVI & Codeplay

- Open invitation to all interested parties to present and discuss
- Collate unfiltered wishlist of potential SYCL SC features

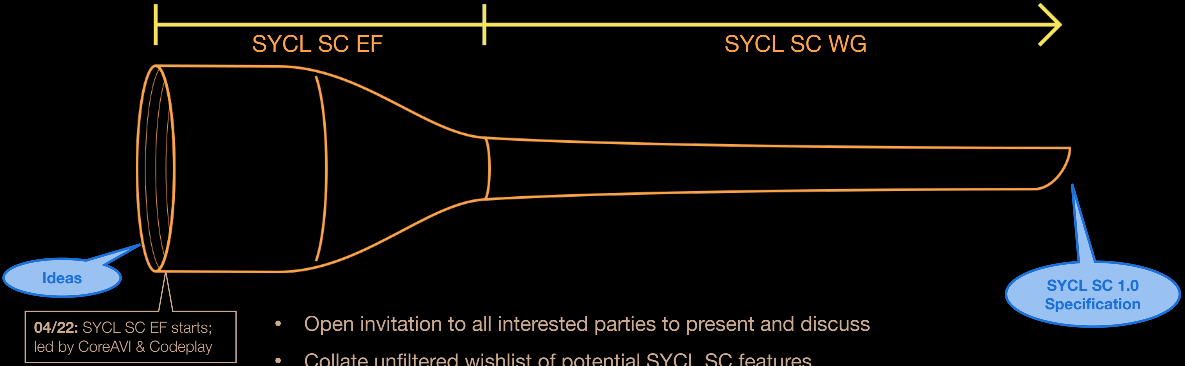
SYCL SC EF: Process



04/22: SYCL SC EF starts; led by CoreAVI & Codeplay

- Open invitation to all interested parties to present and discuss
- Collate unfiltered wishlist of potential SYCL SC features
  - **Goal:** Determine if there is a need for a standard like SYCL SC

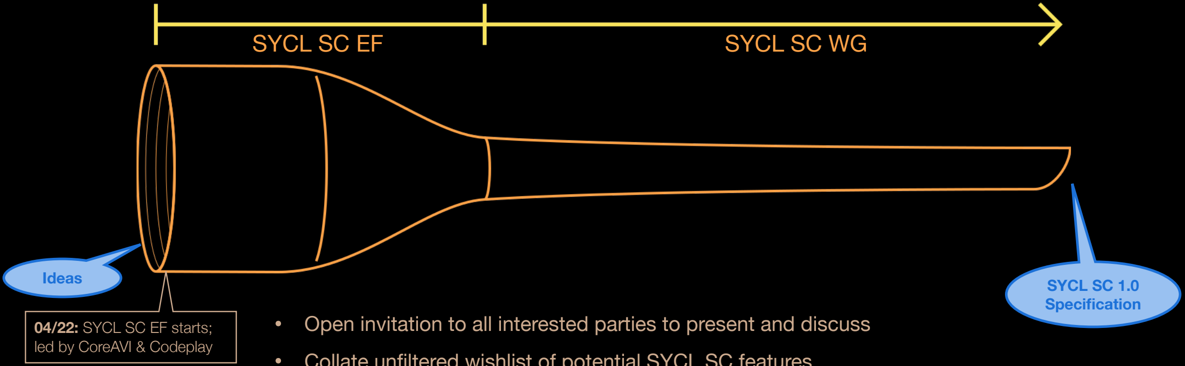
SYCL SC EF: Process



04/22: SYCL SC EF starts; led by CoreAVI & Codeplay

- Open invitation to all interested parties to present and discuss
- Collate unfiltered wishlist of potential SYCL SC features
  - **Goal:** Determine if there is a need for a standard like SYCL SC
  - **Goal:** Understand priorities

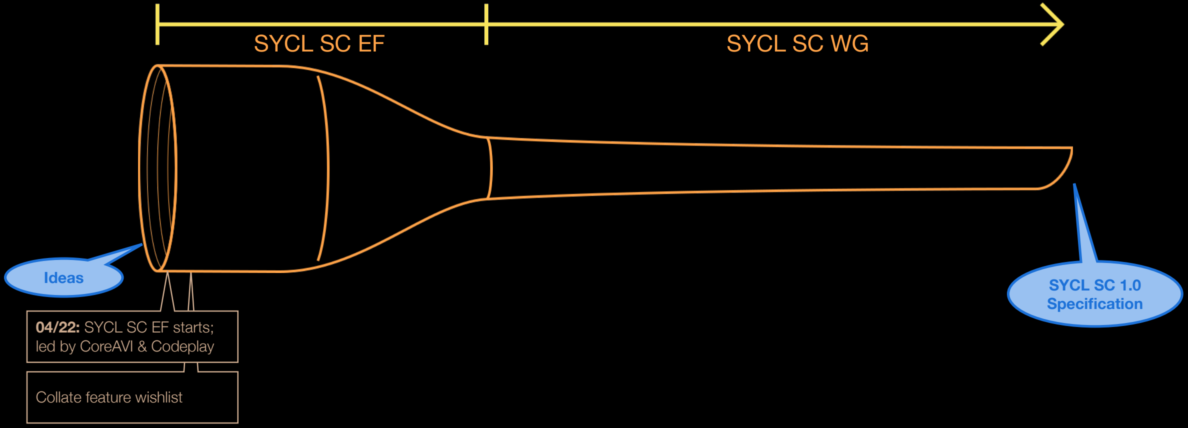
SYCL SC EF: Process



04/22: SYCL SC EF starts; led by CoreAVI & Codeplay

- Open invitation to all interested parties to present and discuss
- Collate unfiltered wishlist of potential SYCL SC features
  - **Goal:** Determine if there is a need for a standard like SYCL SC
  - **Goal:** Understand priorities
  - **Goal:** Identify other requirements

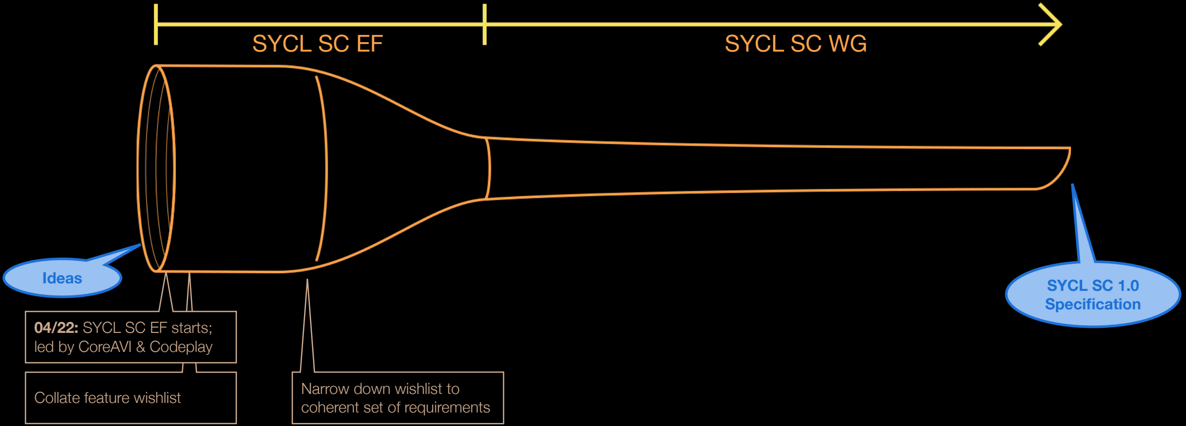
SYCL SC EF: Process



04/22: SYCL SC EF starts; led by CoreAVI & Codeplay

Collate feature wishlist

SYCL SC EF: Process

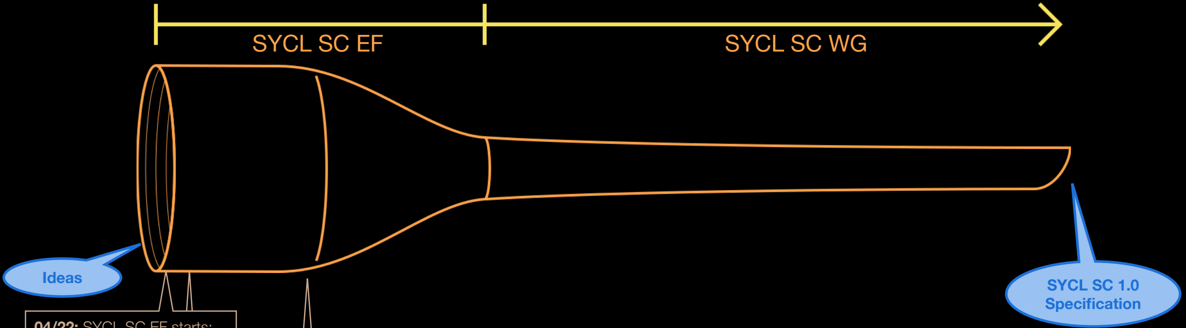


04/22: SYCL SC EF starts; led by CoreAVI & Codeplay

Collate feature wishlist

Narrow down wishlist to coherent set of requirements

SYCL SC EF: Process



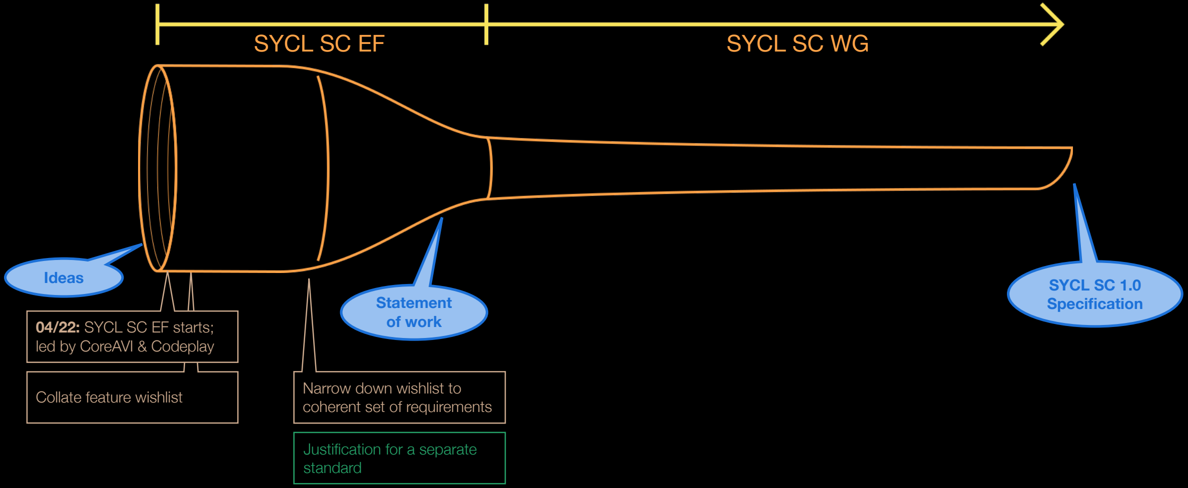
04/22: SYCL SC EF starts; led by CoreAVI & Codeplay

Collate feature wishlist

Narrow down wishlist to coherent set of requirements

Justification for a separate standard

SYCL SC EF: Process



SYCL SC EF

SYCL SC WG

Ideas

04/22: SYCL SC EF starts; led by CoreAVI & Codeplay

Collate feature wishlist

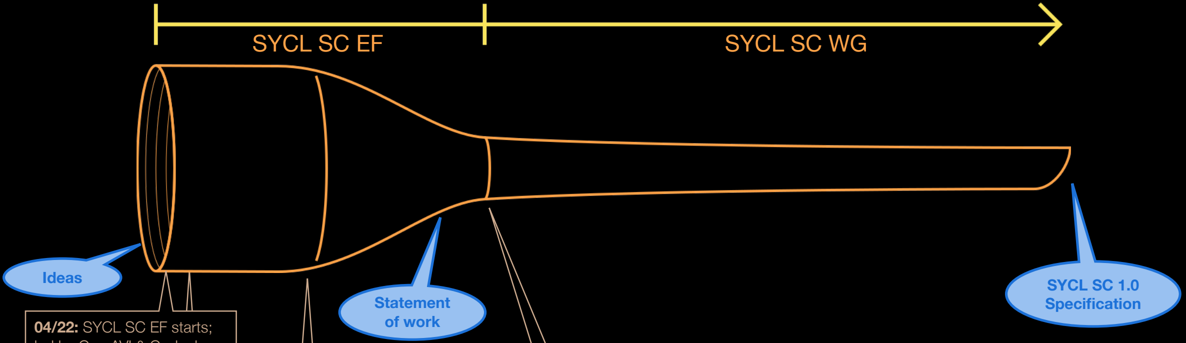
Statement of work

Narrow down wishlist to coherent set of requirements

Justification for a separate standard

SYCL SC 1.0 Specification

SYCL SC EF: Process



04/22: SYCL SC EF starts; led by CoreAVI & Codeplay

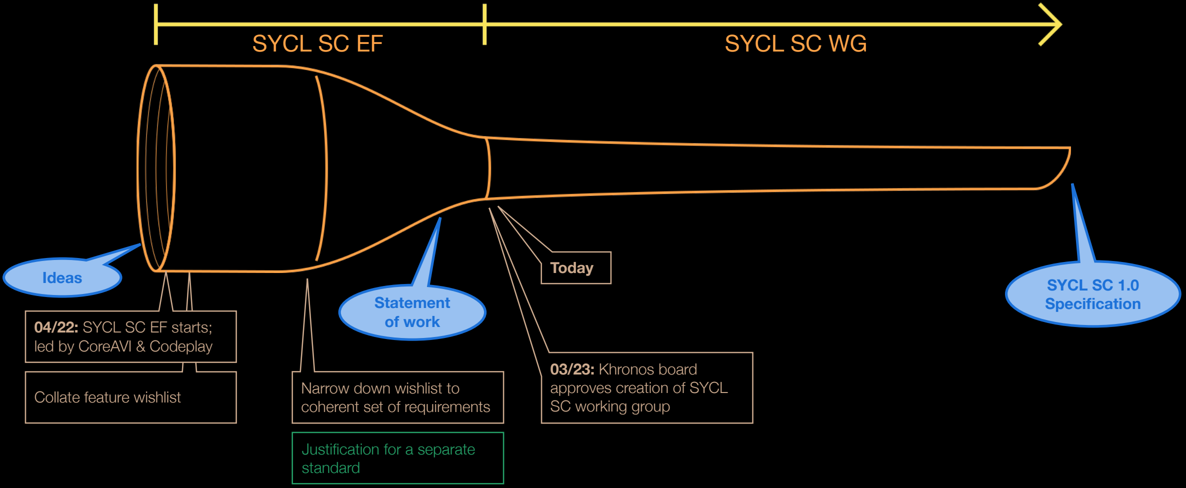
Collate feature wishlist

Narrow down wishlist to coherent set of requirements

Justification for a separate standard

03/23: Khronos board approves creation of SYCL SC working group

SYCL SC EF: Process



04/22: SYCL SC EF starts; led by CoreAVI & Codeplay

Collate feature wishlist

Narrow down wishlist to coherent set of requirements

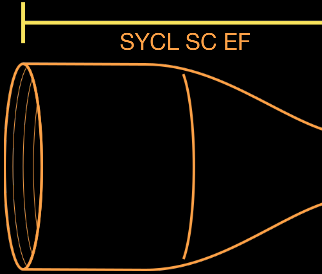
Justification for a separate standard

03/23: Khronos board approves creation of SYCL SC working group

Outline

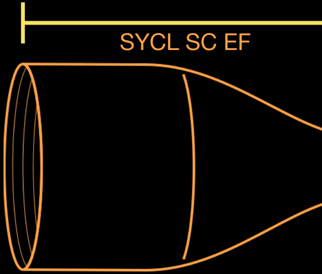
1. Background
2. Summary of SYCL SC EF activities
3. **Technical lessons learned**
   - SYCL SC Scope
   - SC dynamic memory
4. Conclusions

Technical Takeaway: SYCL SC Scope



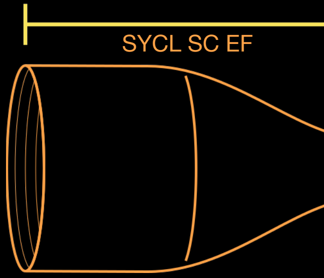
Technical Takeaway: SYCL SC Scope

- Plenty of discussion around the “usual suspects,” like exceptions



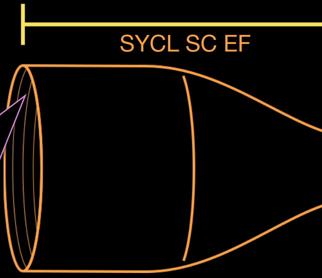
Technical Takeaway: SYCL SC Scope

- Plenty of discussion around the "usual suspects," like exceptions
- Wishlist also contained some more surprising items



Technical Takeaway: SYCL SC Scope

- Plenty of discussion around the "usual suspects," like exceptions
- Wishlist also contained some more surprising items

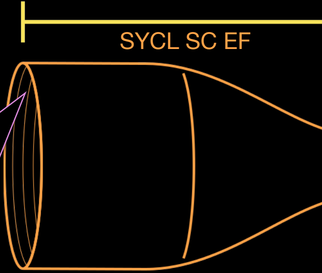


Wishlist:

- …
- Access to accelerators
- …
- Kernel execution on CPU
- …
- Hardware-agnostic
- …
- Other languages? C? Rust?
- …

Technical Takeaway: SYCL SC Scope

- Plenty of discussion around the “usual suspects,” like exceptions
- Wishlist also contained some more surprising items
  - Led to extended discussions

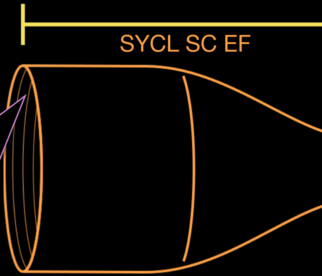


Wishlist:

- …
- Access to accelerators
- …
- Kernel execution on CPU
- …
- Hardware-agnostic
- …
- Other languages? C? Rust?
- …

Technical Takeaway: SYCL SC Scope

- Plenty of discussion around the “usual suspects,” like exceptions
- Wishlist also contained some more surprising items
    - Led to extended discussions
- EF: *Divergence between SYCL and SYCL SC should be minimized*

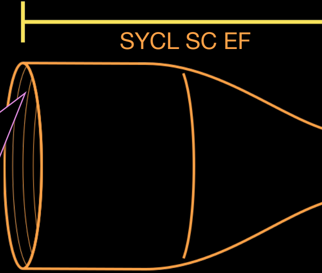


Wishlist:

- …
- Access to accelerators
- …
- Kernel execution on CPU
- …
- Hardware-agnostic
- …
- Other languages? C? Rust?
- …

Technical Takeaway: SYCL SC Scope

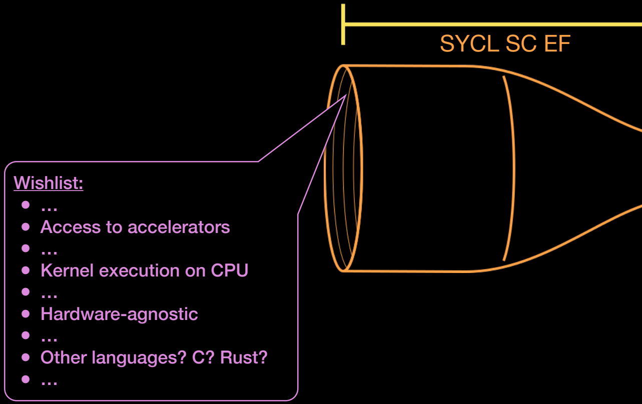
- Plenty of discussion around the “usual suspects,” like exceptions
- Wishlist also contained some more surprising items
    - Led to extended discussions
- EF: *Divergence between SYCL and SYCL SC should be minimized*
    - Features that are essential to SYCL do not need to be called out in the SoW



<u>Wishlist:</u>
- …
- Access to accelerators
- …
- Kernel execution on CPU
- …
- Hardware-agnostic
- …
- Other languages? C? Rust?
- …

Technical Takeaway: SYCL SC Scope

- Plenty of discussion around the “usual suspects,” like exceptions
- Wishlist also contained some more surprising items
  - Led to extended discussions
- EF: *Divergence between SYCL and SYCL SC should be minimized*
  - Features that are essential to SYCL do not need to be called out in the SoW
  - Features that would diverge SYCL SC from SYCL are excluded from the SoW

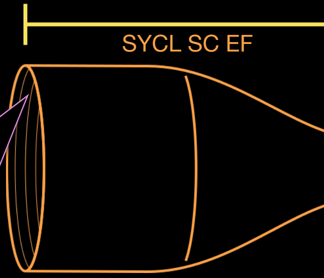


Wishlist:

- …
- Access to accelerators
- …
- Kernel execution on CPU
- …
- Hardware-agnostic
- …
- Other languages? C? Rust?
- …

Technical Takeaway: SYCL SC Scope

- Plenty of discussion around the “usual suspects,” like exceptions
- Wishlist also contained some more surprising items
  - Led to extended discussions
- EF: *Divergence between SYCL and SYCL SC should be minimized*
  - Features that are essential to SYCL do not need to be called out in the SoW
  - Features that would diverge SYCL SC from SYCL are excluded from the SoW
- These wishlist items aren’t in the SYCL SC requirements

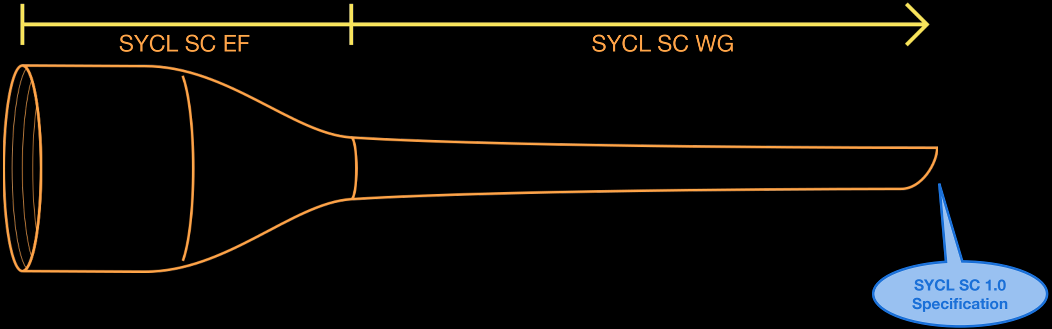


Wishlist:

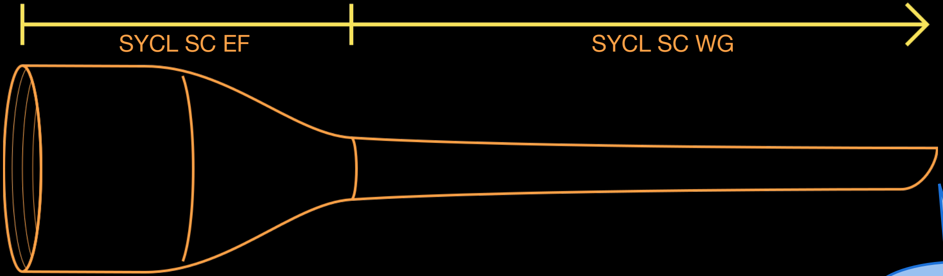
- …
- Access to accelerators
- …
- Kernel execution on CPU
- …
- Hardware-agnostic
- …
- Other languages? C? Rust?
- …

Technical Takeaway: Dynamic Memory

Technical Takeaway: Dynamic Memory



Technical Takeaway: Dynamic Memory



Wishlist:

- …
- (No) dynamic memory
- …

SYCL SC 1.0 Specification

Technical Takeaway: Dynamic Memory

Technical Takeaway: Dynamic Memory

- Need to cut through FUD

Technical Takeaway: Dynamic Memory

- Need to cut through FUD
- **Q:** What, precisely, is bad about dynamic memory?

Technical Takeaway: Dynamic Memory

- Need to cut through FUD
- **Q:** What, precisely, is bad about dynamic memory?
- **A:** Non-determinism

Technical Takeaway: Dynamic Memory

- Need to cut through FUD
- **Q:** What, precisely, is bad about dynamic memory?
- **A:** Non-determinism
  - `malloc()` crosses from user space into the OS

Technical Takeaway: Dynamic Memory

- Need to cut through FUD
- **Q:** What, precisely, is bad about dynamic memory?
- **A:** Non-determinism
  - `malloc()` crosses from user space into the OS
  - `malloc()` takes an unpredictable amount of time to find memory

Technical Takeaway: Dynamic Memory

- Need to cut through FUD
- **Q:** What, precisely, is bad about dynamic memory?
- **A:** Non-determinism
  - `malloc()` crosses from user space into the OS
  - `malloc()` takes an unpredictable amount of time to find memory
  - `malloc()` is allowed to return `NULL`

Technical Takeaway: Dynamic Memory

- Need to cut through FUD
- **Q:** What, precisely, is bad about dynamic memory?
- **A:** Non-determinism
  - `malloc()` crosses from user space into the OS
  - `malloc()` takes an unpredictable amount of time to find memory
  - `malloc()` is allowed to return `NULL`
- Safety-critical applications have strict deadlines

Technical Takeaway: Dynamic Memory

- Need to cut through FUD
- **Q:** What, precisely, is bad about dynamic memory?
- **A:** Non-determinism
  - `malloc()` crosses from user space into the OS
  - `malloc()` takes an unpredictable amount of time to find memory
  - `malloc()` is allowed to return `NULL`
- Safety-critical applications have strict deadlines
  - If deadlines can be missed in unpredictable ways, then safety certification becomes difficult

Technical Takeaway: Dynamic Memory

- Need to cut through FUD
- **Q:** What, precisely, is bad about dynamic memory?
- **A:** Non-determinism
  - `malloc()` crosses from user space into the OS
  - `malloc()` takes an unpredictable amount of time to find memory
  - `malloc()` is allowed to return `NULL`
- Safety-critical applications have strict deadlines
  - If deadlines can be missed in unpredictable ways, then safety certification becomes difficult
- Problem isn't dynamic memory; problem is non-determinism

Technical Takeaway: Dynamic Memory

Technical Takeaway: Dynamic Memory

- Dynamic memory can be made deterministic:

Technical Takeaway: Dynamic Memory

- Dynamic memory can be made deterministic:
- **Solution:** Ensure all calls to `malloc()` happen before application enters time-critical phase

Technical Takeaway: Dynamic Memory

- Dynamic memory can be made deterministic:
- **Solution:** Ensure all calls to `malloc()` happen before application enters time-critical phase
- **Solution:** Don’t call `free()`

Technical Takeaway: Dynamic Memory

- Dynamic memory can be made deterministic:
- **Solution:** Ensure all calls to `malloc()` happen before application enters time-critical phase
- **Solution:** Don't call `free()`
  - Avoids most complexity with `malloc()`

Technical Takeaway: Dynamic Memory

- Dynamic memory can be made deterministic:
- **Solution:** Ensure all calls to `malloc()` happen before application enters time-critical phase
- **Solution:** Don't call `free()`
  - Avoids most complexity with `malloc()`
  - Only possible for some applications

Technical Takeaway: Dynamic Memory

- Dynamic memory can be made deterministic:
- **Solution:** Ensure all calls to `malloc()` happen before application enters time-critical phase
- **Solution:** Don’t call `free()`
  - Avoids most complexity with `malloc()`
  - Only possible for some applications
- **Solution:** Use object pools

Technical Takeaway: Dynamic Memory

- Dynamic memory can be made deterministic:
- **Solution:** Ensure all calls to `malloc()` happen before application enters time-critical phase
- **Solution:** Don’t call `free()`
  - Avoids most complexity with `malloc()`
  - Only possible for some applications
- **Solution:** Use object pools
  - Hard upper bound for number of objects

Technical Takeaway: Dynamic Memory

- Dynamic memory can be made deterministic:
- **Solution:** Ensure all calls to `malloc()` happen before application enters time-critical phase
- **Solution:** Don't call `free()`
  - Avoids most complexity with `malloc()`
  - Only possible for some applications
- **Solution:** Use object pools
  - Hard upper bound for number of objects
  - Vulkan SC solution

Technical Takeaway: Dynamic Memory

- Dynamic memory can be made deterministic:
- **Solution:** Ensure all calls to `malloc()` happen before application enters time-critical phase
- **Solution:** Don't call `free()`
  - Avoids most complexity with `malloc()`
  - Only possible for some applications
- **Solution:** Use object pools
  - Hard upper bound for number of objects
  - Vulkan SC solution
- **Solution:** Use local (stack) buffers

Technical Takeaway: Dynamic Memory

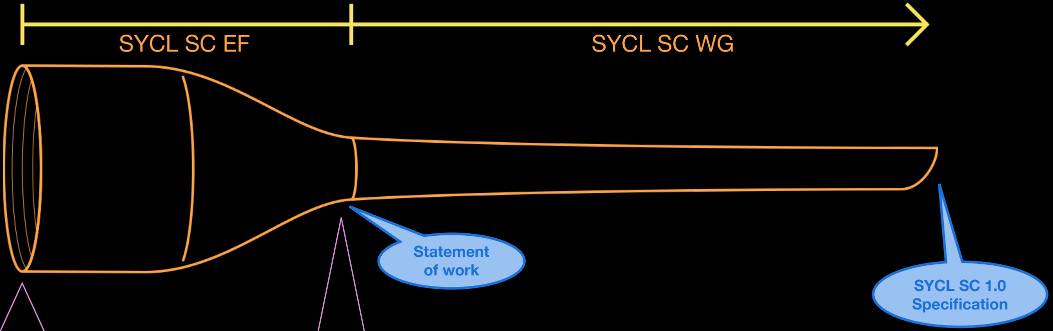
- Dynamic memory can be made deterministic:
- **Solution:** Ensure all calls to `malloc()` happen before application enters time-critical phase
- **Solution:** Don't call `free()`
  - Avoids most complexity with `malloc()`
  - Only possible for some applications
- **Solution:** Use object pools
  - Hard upper bound for number of objects
  - Vulkan SC solution
- **Solution:** Use local (stack) buffers
- **Solution:** Others?

Technical Takeaway: Dynamic Memory

- Dynamic memory can be made deterministic:
- **Solution:** Ensure all calls to `malloc()` happen before application enters time-critical phase
- **Solution:** Don’t call `free()`
  - Avoids most complexity with `malloc()`
  - Only possible for some applications
- **Solution:** Use object pools
  - Hard upper bound for number of objects
  - Vulkan SC solution
- **Solution:** Use local (stack) buffers
- **Solution:** Others?

- **Conclusion:** Deterministic memory management is possible

Technical Takeaway: Dynamic Memory



Wishlist:

- ...
- (No) dynamic memory
- ...

Requirements:

- ...
- Deterministic memory management
- ...

Technical Takeaway: Dynamic Memory

Technical Takeaway: Dynamic Memory

- Many issues like dynamic memory

Technical Takeaway: Dynamic Memory

- Many issues like dynamic memory
- SYCL SC WG will need to…

Technical Takeaway: Dynamic Memory

- Many issues like dynamic memory
- SYCL SC WG will need to…
  - Cut through uncertainty to understand underlying safety concern

Technical Takeaway: Dynamic Memory

- Many issues like dynamic memory
- SYCL SC WG will need to…
  - Cut through uncertainty to understand underlying safety concern
  - Identify possible solutions

Technical Takeaway: Dynamic Memory

- Many issues like dynamic memory
- SYCL SC WG will need to…
    - Cut through uncertainty to understand underlying safety concern
    - Identify possible solutions
    - Make changes to SYCL spec so that SYCL SC is compatible with solutions

Technical Takeaway: Dynamic Memory

- Many issues like dynamic memory
- SYCL SC WG will need to…
  - Cut through uncertainty to understand underlying safety concern
  - Identify possible solutions
  - Make changes to SYCL spec so that SYCL SC is compatible with solutions
- As much as possible, SYCL SC will not mandate a specific solution

Outline

1. Background
2. Summary of SYCL SC EF activities
3. Technical lessons learned
   - SYCL SC Scope
   - SC dynamic memory
4. **Conclusions**

Call to Action

Call to Action

- Summary
  - What is SC
  - Need for SYCL SC
  - SYCL SC EF overview
  - Technical Takeaways

Call to Action

- Summary
  - What is SC
  - Need for SYCL SC
  - SYCL SC EF overview
  - Technical Takeaways
- Join the SYCL SC working group

Call to Action

- Summary
  - What is SC
  - Need for SYCL SC
  - SYCL SC EF overview
  - Technical Takeaways
- Join the SYCL SC working group
  - Join Khronos

Call to Action

- Summary
  - What is SC
  - Need for SYCL SC
  - SYCL SC EF overview
  - Technical Takeaways
- Join the SYCL SC working group
  - Join Khronos
- Planning an advisory panel for Khronos non-members

Call to Action

- Summary
  - What is SC
  - Need for SYCL SC
  - SYCL SC EF overview
  - Technical Takeaways
- Join the SYCL SC working group
  - Join Khronos
- Planning an advisory panel for Khronos non-members
  - Get in touch if interested

Call to Action

- Summary
  - What is SC
  - Need for SYCL SC
  - SYCL SC EF overview
  - Technical Takeaways
- Join the SYCL SC working group
  - Join Khronos
- Planning an advisory panel for Khronos non-members
  - Get in touch if interested
- Just talk to us

The 11th International workshop on OpenCL and SYCL

IWOCL & SYCLcon 2023

What’s New in SYCL SC™ (SYCL for Safety Critical Systems)

Erik Tomusk, Codeplay Software erik@codeplay.com

Verena Beckham, Codeplay Software verena@codeplay.com

April 18-20, 2023 | University of Cambridge, UK

iwocl.org