SYCL v1.2 release

IWOCL, May 2015

3D Incorporated · Adobe · ALTERA · amazon.com · ARCANE Technologies · AXELL CORPORATION · AXIS COMMUNICATIONS · BLIZZARD ENTERTAINMENT · BROADCOM · cadence

CANONICAL · CEVA · codeplay · cognivue Innovating With Vision · Columbia University · Continental · COREAVI · DASSAULT SYSTEMES

DMP · EA · ENTROPIC · ERICSSON · ETRI Electronics and Telecommunications Research Institute · freescale semiconductor · FUTUREMARK CORPORATION · Gaijin entertainment · Google · harmonic · HUAWEI

IBM · Imperial College London · 財團法人資訊工業策進會 INSTITUTE FOR INFORMATION INDUSTRY · ITRI Industrial Technology Research Institute · itseez · KDAB · KISHONTI INFORMATICS · KNU KYUNGPOOK NATIONAL UNIVERSITY · KOREA UNIVERSITY · LG

BOARD OF PROMOTERS

AMD · ARM · Apple · SONY · SAMSUNG · EPIC GAMES · intel · VIVANTE · NOKIA · NVIDIA · Imagination · QUALCOMM

KHRONOS™ GROUP

Over 100 members worldwide
any company is welcome to join

Linaro · LUCASFILM Ltd · MARVELL · matrox · MEDIATEK · Mentor Graphics · Microsoft · MIT Lincoln Laboratory · mobica · Movidius · mozilla

MULTICOREWARE · National Taiwan University · 日本大学 · NEC · OSU Oregon State University · Oculus VR · OXIDE · Panasonic · PIXAR · POLITECNICO DI MILANO · RENESAS

RICOH imagine. change. · RIGHTWARE · 서울대학교 SEOUL NATIONAL UNIVERSITY · smithmicro software · SPREADTRUM · ST life.augmented · symbio · SYNOPSYS · TAKUMI · TAMPERE UNIVERSITY OF TECHNOLOGY

TES Electronic Solutions · Texas Instruments · THINCI · Think Silicon · tobii · TOSHIBA · TRANSGAMING · unity · ALMA MATER STUDIORUM A.D. 1088 · University of BRISTOL · UNIVERSITY OF CAMBRIDGE

VALVE · VeriSilicon · VIA we connect · videantis passion for video · Visteon · vmware · WARGAMING.NET LET'S BATTLE · XILINX · zSpace

SYCL is not magic

SYCL is a practical, open, royalty-free standard to deliver high performance software on today's highly-parallel systems

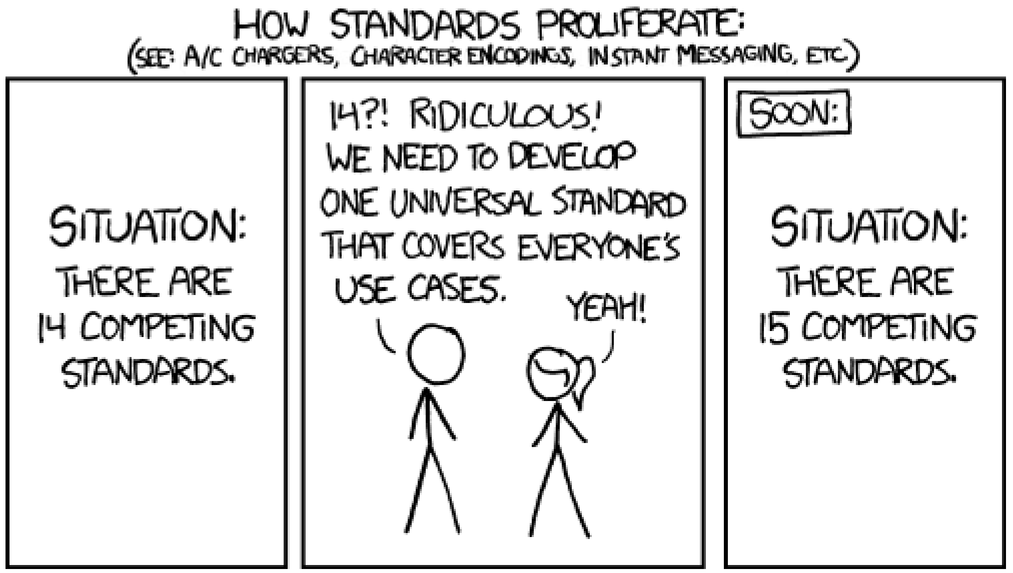
What is SYCL for?

- **Modern C++ lets us separate the what from the how :**
  - We want to separate **what** the user wants to do: *science, computer vision, AI …*
  - And enable the **how** to be: *run fast on an OpenCL device*
- **Modern C++ supports and encourages this separation**

What we want to achieve

- **We want to enable a C++ ecosystem for OpenCL:**
  - C++ template libraries
  - Tools: compilers, debuggers, IDEs, optimizers
  - Training, example programs
  - Long-term support for current and future OpenCL features

Why a new standard?


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

http://imgs.xkcd.com/comics/standards.png

- **There are already very established ways to map C++ to parallel processors**
  - So we follow the established approaches
- **There are specifics to do with OpenCL we need to map to C++**
  - We have worked hard to be an *enabler* for other C++ parallel standards
- **We add no more than we need to**

What features of OpenCL do we need?

- **We want to enable all OpenCL features in C++ with SYCL**
  - Images, work-groups, barriers, constant/global/local/private memory
  - Memory sharing: mapping and DMA
  - Platforms, contexts, events, queues
  - Support wide range of OpenCL devices: CPUs, GPUs, FPGAs, DSPs…
- **We want to make it easy to write high-performance OpenCL code in C++**
  - SYCL code in C++ must use memory and execute kernels efficiently
  - We must provide developers with all the optimization options they have in OpenCL
- **We want to enable OpenCL C code to interoperate with C++ SYCL code**
  - Sharing of contexts, memory objects etc

How do we bring OpenCL features to C++?

- **Key decisions:**
  - We will not add any language extensions to C++
  - We will work with existing C++ compilers
  - We will provide the full OpenCL feature-set in C++

How did we come to our decisions?

What was our thinking?

Single-source vs C++ kernel language

- **Single-source: a single-source file contains both host and device code**
  - Type-checking between host and device
  - A single template instantiation can create all the code to kick off work, manage data and execute the kernel
    - **e.g. sort<MyClass> (myData);**
  - The approach taken by C++ 17 Parallel STL as well as SYCL

- **C++ kernel language**
  - Matches standard OpenCL C
  - Proposed for OpenCL v2.1
  - Being considered as an addition for SYCL v2.1

Why ‘name’ kernels?

- **Enables implementers to have multiple, different compilers for host and different devices**
  - With SYCL, software developers can choose to use the best compiler for CPU and the best compiler for each individual device they want to support
  - The resulting application will be highly optimized for CPU *and* OpenCL devices
  - Easy-to-integrate into existing build systems

- **Only required for C++11 lambdas, not required for C++ *functors***
  - Required because lambdas don’t have a name to enable linking between different compilers

Buffers/images/accessors vs shared pointers

- **OpenCL v1.2 supports a wide range of different devices and operating systems**
  - All shared data must be encapsulated in OpenCL memory objects: buffers and images
  - To enable SYCL to achieve maximum performance of OpenCL, we follow OpenCL's memory model approach
  - But, we apply OpenCL's memory model to C++ with buffers, images and accessors
  - Separation of data storage and data access

Hierarchical parallelism

- A whole new approach
- Enables high-performance, portable C++ template algorithms to work across CPUs, GPUs and other devices easily
- Is really just syntactical

What can I do with SYCL?

Anything you can do with C++!

With the performance and portability of OpenCL

Progress report on the SYCL vision

- ✓ Open, royalty-free standard: released
- ✓ Conformance testsuite: going into adopters package

- ➢ Open-source implementation: in progress (triSYCL)
- ➢ Commercial, conformant implementation: in progress
- ➢ C++ 17 Parallel STL: open-source in progress

- Template libraries for important C++ algorithms: getting going
- Integration into existing parallel C++ libraries: getting going

Building the SYCL for OpenCL ecosystem

- **To deliver on the full potential of high-performance heterogeneous systems**
  - We need the libraries
  - We need integrated tools
  - We need implementations
  - We need training and examples
- **An open standard makes it much easier for people to work together**
  - SYCL is a group effort
  - We have designed SYCL for maximum ease of integration

Questions

And maybe some volunteering of joining in to build the ecosystem?

☺