OpenCL™ “SPIR” a Standard Portable IR

Boaz Ouriel, Intel

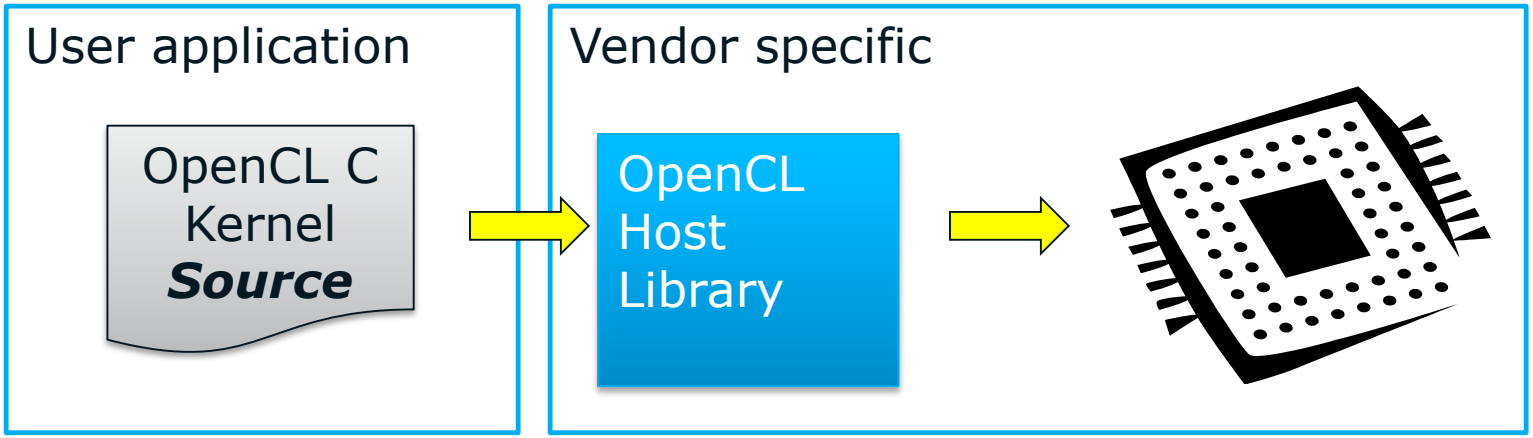
Compiler, Architecture and Tools Conference

Nov. 2013

SPIR = Standard Portable Intermediate Representation

- A portable non-source representation for OpenCL 1.2 device programs
- A Khronos initiative
  - SPIR 1.2 preview spec standardizes *consumption*
  - http://www.khronos.org/files/opencl-spir-12-provisional.pdf
- Based on LLVM 3.2
- Open source *generator*: Clang

OpenCL: Source compilation flow

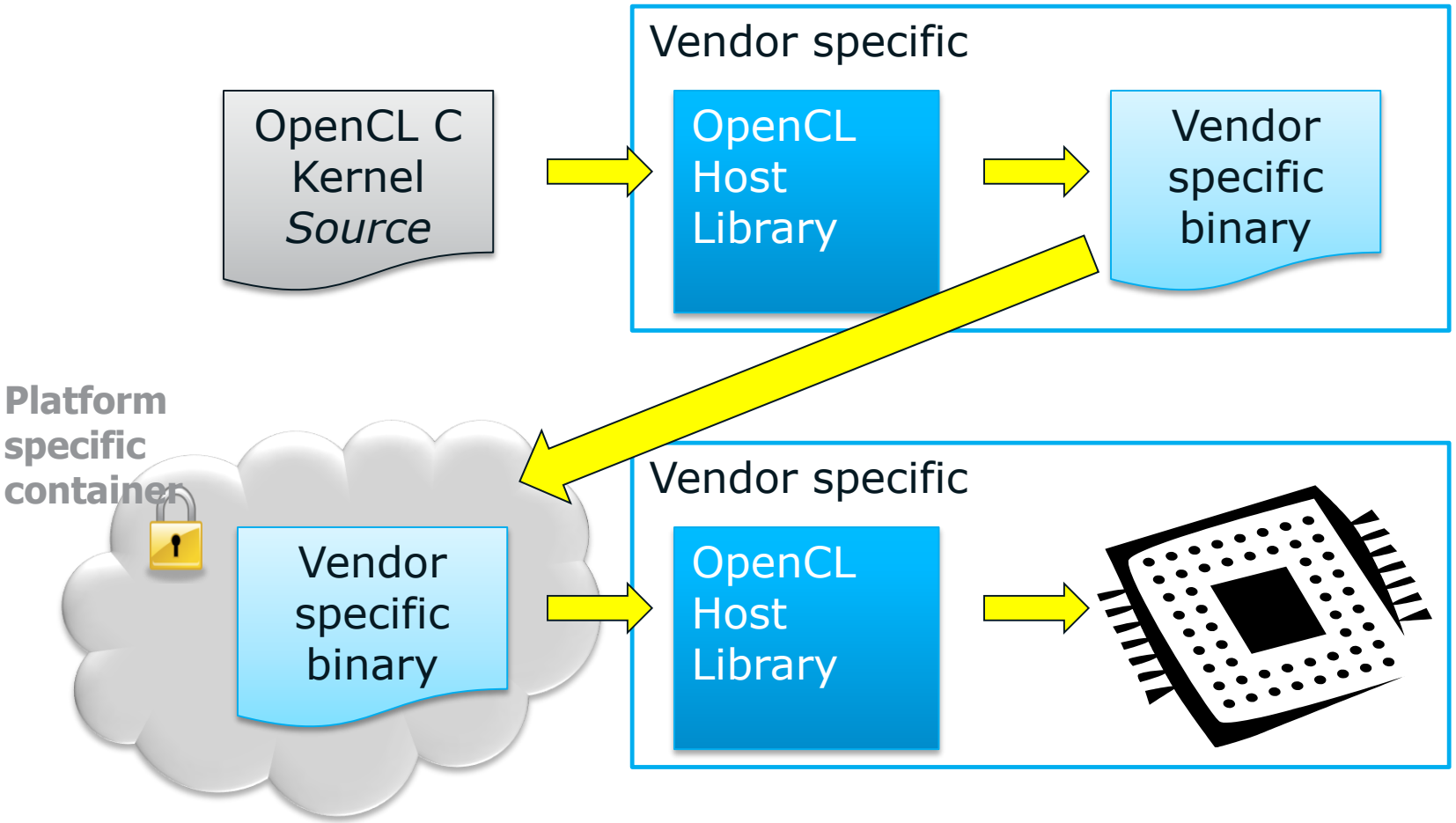


ISV ships their kernel source

- Exposes their IP

Supports only OpenCL C

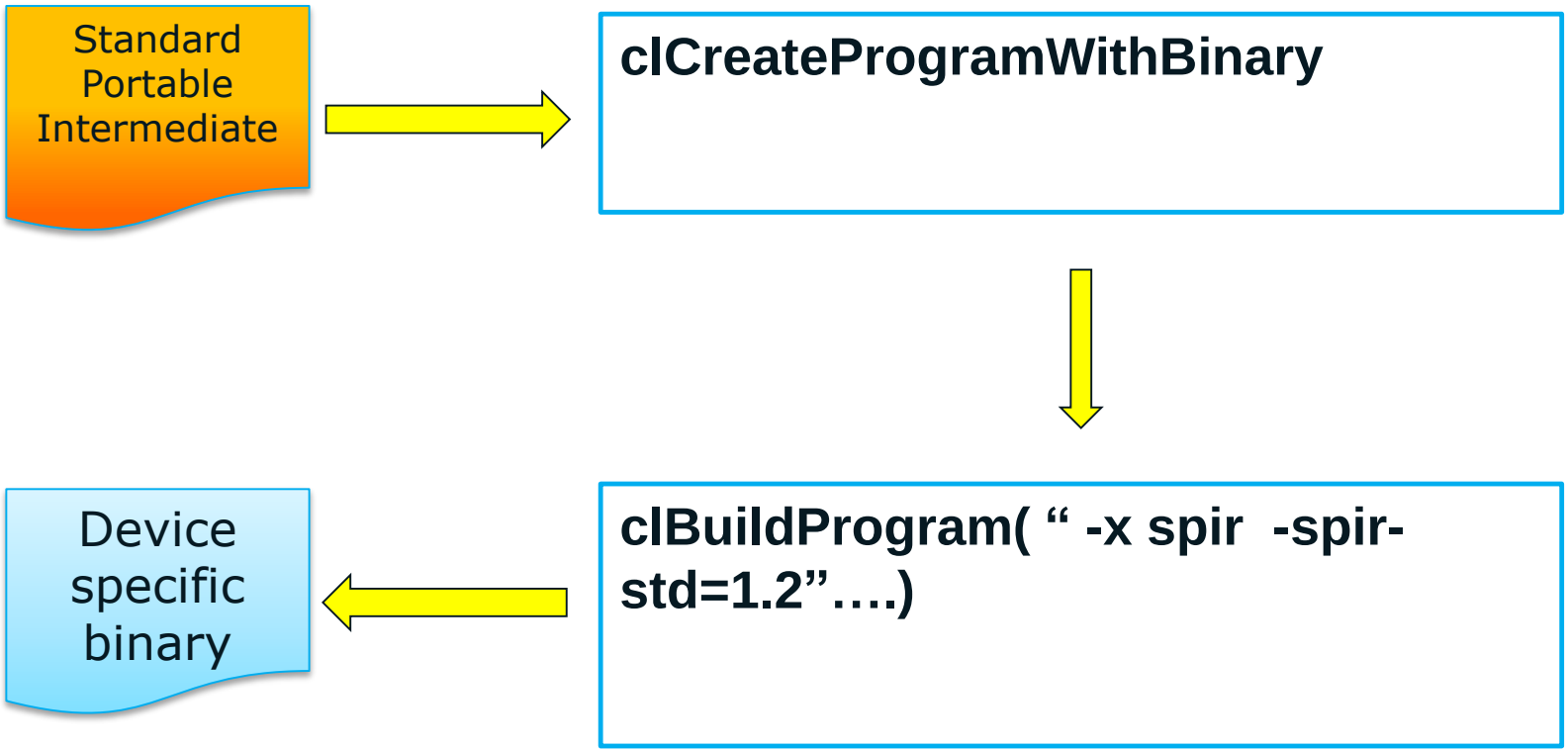
OpenCL: Binary compilation flow



ISV ships vendor-specific binary

- Proliferation: devices, driver revisions, vendors
- Market-lagging: target *shipped* products

Sample SPIR consumption flow



SPIR Goals

Reduce ISV pain

- Avoid IP exposure: do not ship source
- Manage device/driver/vendor proliferation
- Avoid market lag

Open a door for innovation on top of OpenCL™ standard

- Support 3rd party compilers
- Allow additional programming languages to target OpenCL runtimes

LLVM IR

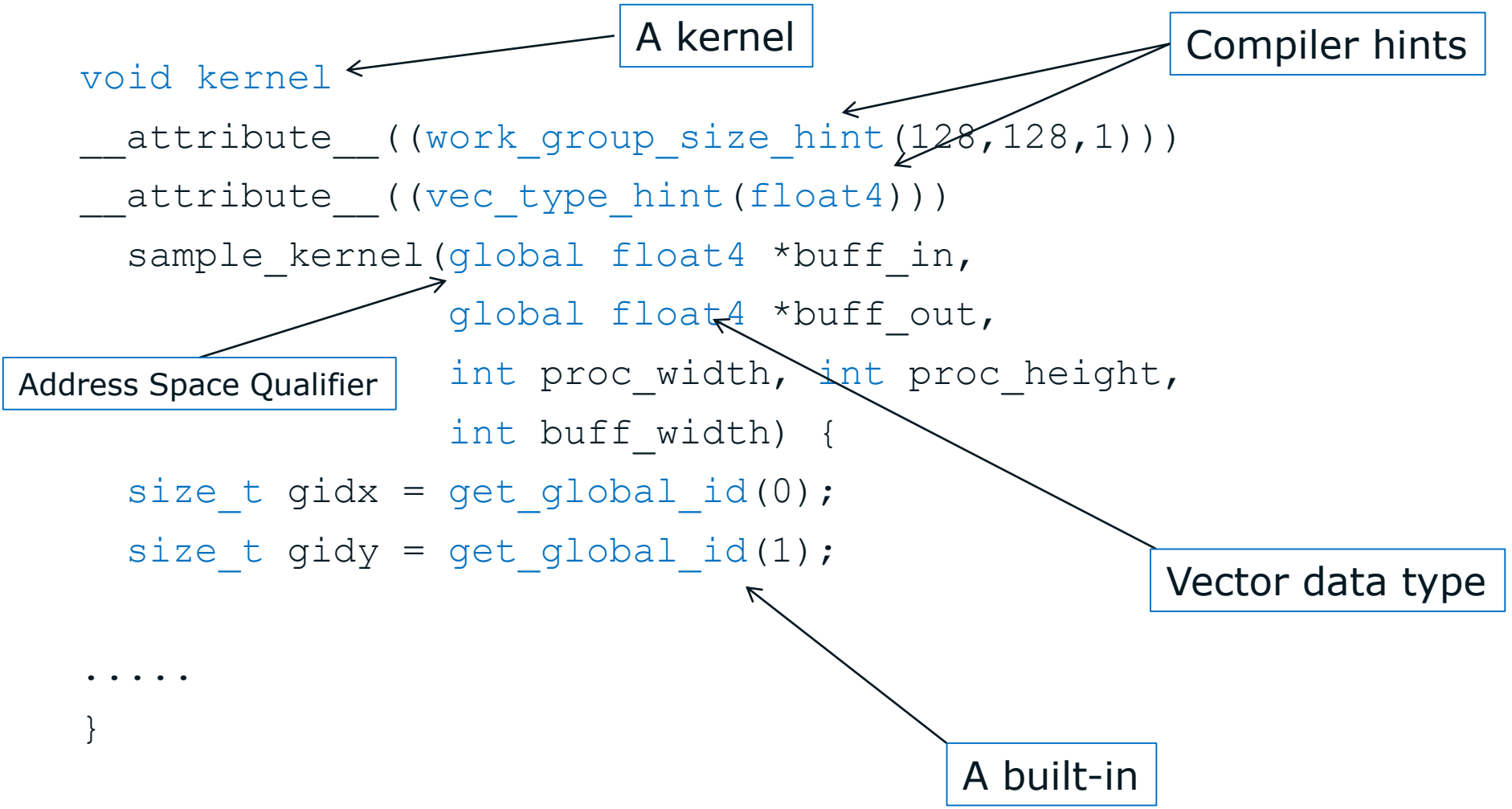
- Static Single Assignment (SSA) based representation
- Strong typed, low-level operations
- capable of representing high-level languages cleanly
- used throughout all phases of LLVM mid-end optimizations

```
define i32 @mul_add(i32 %x, i32 %y, i32 %z) {
entry:
  %tmp = mul i32 %x, %y
  %tmp2 = add i32 %tmp, %z
  ret i32 %tmp2
}
```

A Closer Look at SPIR

- Layered on top of LLVM IR
  - A subset of LLVM IR
    - Example: LLVM *i3* data type is illegal by SPIR specification
  - SPIR related changes are available at llvm.org since llvm 3.0
    - New SPIR32 and SPIR 64 targets
    - New SPIR function and SPIR kernel calling conventions
- Standardizes the LLVM representation of OpenCL™ programs
  - Data types, built-ins name mangling, ABI, and so on…
  - OpenCL-specific information
    - Utilizes LLVM metadata infrastructure
    - Examples: OpenCL standard version, SPIR version, compiler options
- A SPIR version is associated with an LLVM version

Example: a cl program anatomy




```
void kernel
__attribute__((work_group_size_hint(128,128,1)))
__attribute__((vec_type_hint(float4)))
  sample_kernel(global float4 *buff_in,
                global float4 *buff_out,
                int proc_width, int proc_height,
                int buff_width) {
  size_t gidx = get_global_id(0);
  size_t gidy = get_global_id(1);

.....
}
```

Example: the SPIR binary anatomy

```
target datalayout = ...
target triple = "spir-unknown-unknown"
define spir_kernel
void
@sample_kernel(<4 x float> addrspace(1)* nocapture %buff_in,
               <4 x float> addrspace(1)* nocapture %buff_out,
                i32 %proc_width, i32 %proc_height,
                i32 %buff_width) nounwind
{
entry:
%call = tail call spir_func i32 @_Z13get_global_idj(i32 0) nounwind readnone
. . .
ret void
}
```

SPIR Target Information

A Kernel CC

built-ins name mangling

A built-in / user function CC

OpenCL specific metadata

```
!1 = metadata !{metadata !"work_group_size_hint", i32 128, i32 128, i32 1}
!2 = metadata !{metadata !"vec_type_hint", <4 x float> undef, i32 0}
...
```

SPIR Usage Advantages

Standardizes a portable binary format

- Brings functional portability to OpenCL™ programs at binary level

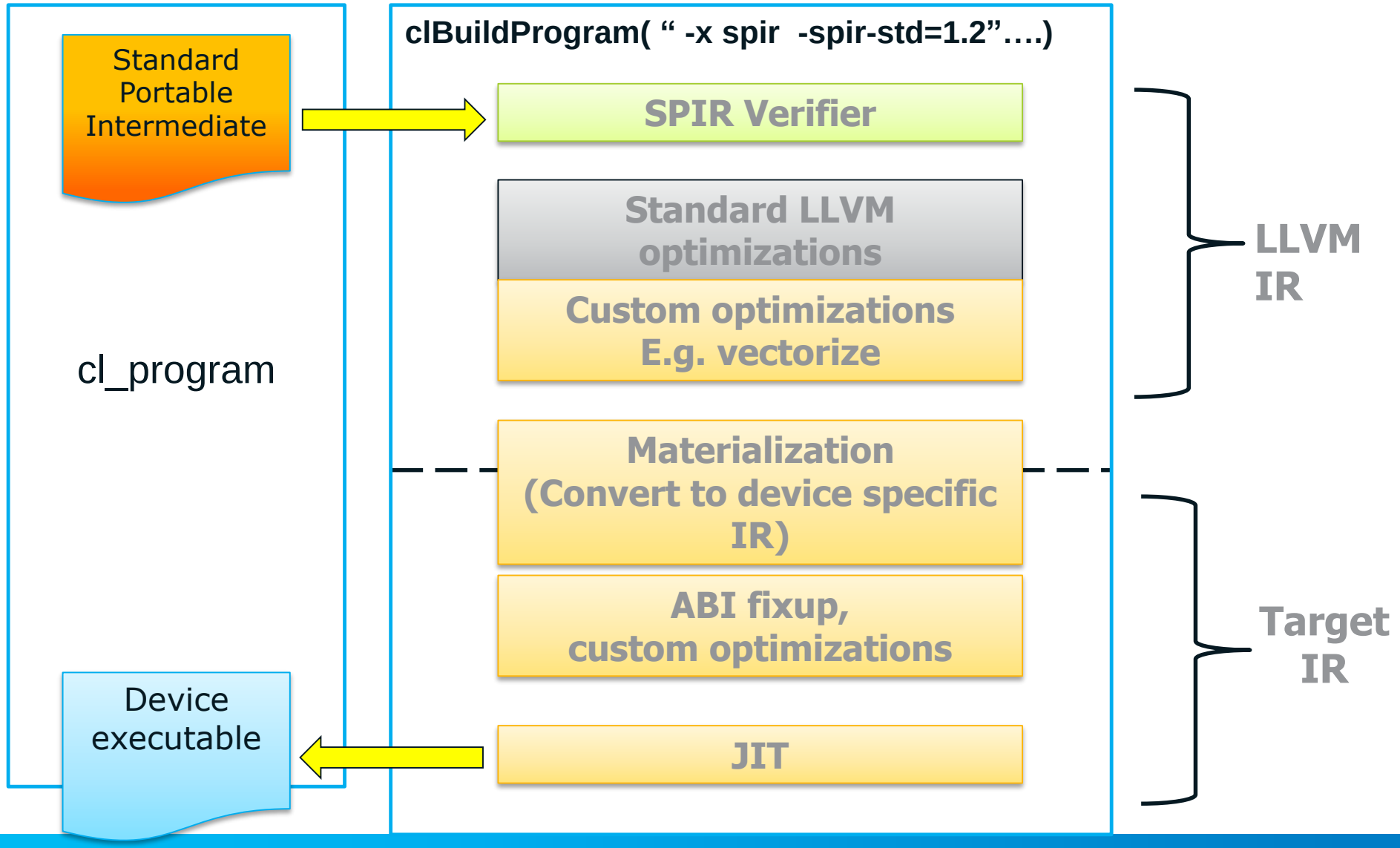
Enable additional high-level languages to target OpenCL implementations

- Simplifies cross-platform enablement
- An industry-open standard

Layered on top of LLVM

- Open source with permissive license
- Well-known & well-documented compiler infrastructure
- Clang includes most of the changes to support SPIR generation

Sample SPIR flow: Room for optimizations



Shevlin Park as a SPIR Generator

- A C++ AMP implementation in Clang/LLVM using OpenCL™ standard
  - Presented at LLVM November 2012 developer conference
    - By Dillon Sharlet from Intel
  - http://llvm.org/devmtg/2012-11/Sharlet-ShevlinPark.pdf
- Translate C++ AMP kernel code to OpenCL "C" programs
- Would rather target SPIR instead of generating OpenCL "C"
  - Examples:
    - Save redundant unstructured to structured control flow hassle
    - Better code quality due to less reliance on runtime optimizations
    - More efficient device code compilation due to transferring compilation cost from runtime to compile time

SPIR 1.2 support by Intel

Intel® SDK for OpenCL* Applications XE 2013 introduces SPIR 1.2 as a preview feature

- Download link: http://software.intel.com/en-us/vcsource/tools/opencl-sdk-xe
- Announcement link: http://software.intel.com/en-us/forums/topic/475429

Go Ahead and give it a try ☺

Thank you

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

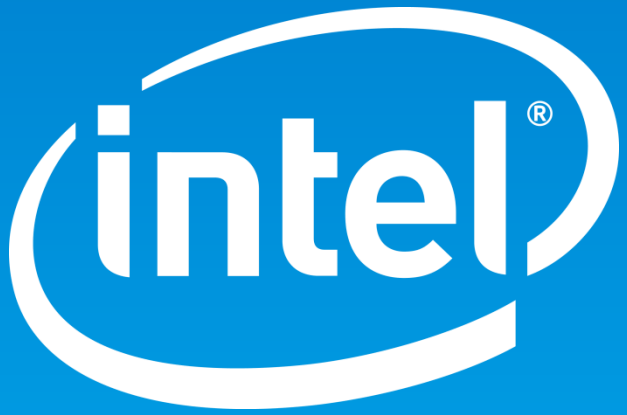
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that

VTune, and Cilk are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804


intel