GPU DAEMON
ROAD TO ZERO COST SUBMISSION

MICHAL MROZEK

ZBIGNIEW ZDANOWICZ

Legal Notices and Disclaimers

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information. The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

- 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 product when combined with other products.
- All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.
- All products, platforms, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice. All dates specified are target dates, are provided for planning purposes only and are subject to change.

- This document contains information on products in the design phase of development. Do not finalize a design with this information. Revised information will be published when the product is available. Verify with your local sales office that you have the latest datasheet before finalizing a design.
- Code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user.
- Intel, Intel Inside, Intel Atom and Intel Core are trademarks of Intel Corporation in the U.S. and other countries.
- Other names and brands may be claimed as the property of others.
- Copyright © 2015-2016, Intel Corporation. All rights reserved.
- OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Agenda

- Current OpenCL™ scheduling model
- GPU Daemon:
  - Instant Mode
  - Enqueue Mode
- Performance Data
- Efficient use of GPU Daemon patterns
- Summary

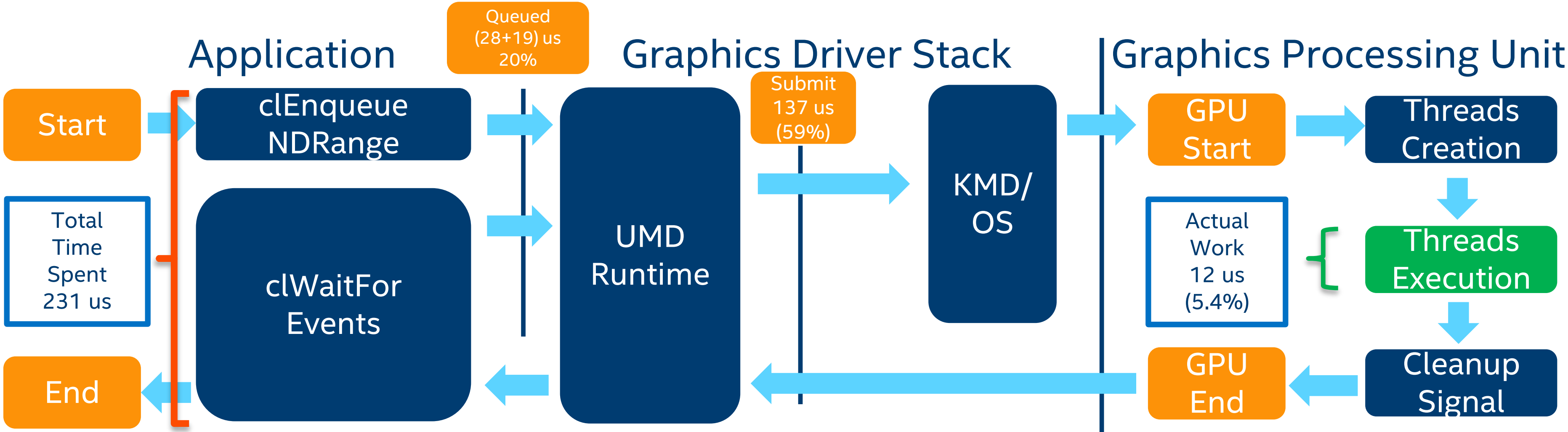
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Agenda

- Current OpenCL™ scheduling model
- GPU Daemon:
  - Instant Mode
  - Enqueue Mode
- Performance Data
- Efficient use of GPU Daemon patterns
- Summary

Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Current OpenCL™ scheduling model



Submission latencies	CPU Start to Queue	Queue to Submit	Submit to GPU Start	GPU Start to GPU End REAL WORK	GPU End To CPU End	Total
Subsequent enqueue	28	19	137	12	32	231

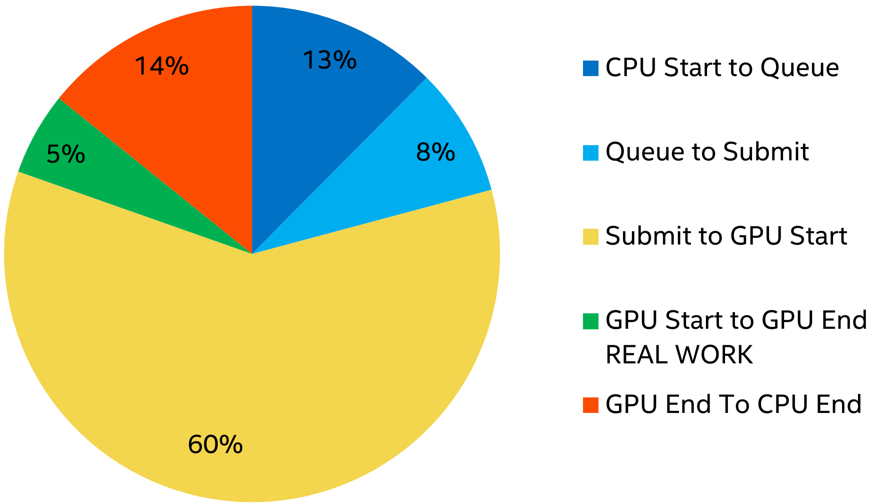
intel
Iris™ Pro
Graphics

Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Current OpenCL™ scheduling model

Submission latencies	CPU Start to Queue	Queue to Submit	Submit to GPU Start	GPU Start to GPU End REAL WORK	GPU End To CPU End	Total
Subsequent enqueue	28	19	137	12	32	231



- Driver overhead is significant:
  - Not suitable for small kernels.
  - Not suitable for low latency scenarios.
- Submission is expensive:
  - Memory needs to be resident.
  - GPU threads are created & destroyed for each kernel.
- Why queue if I want to submit ?
  - No queue needed if 0 cost submission & completion.

Current scheduling model doesn't suit well for low latency / short workloads

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

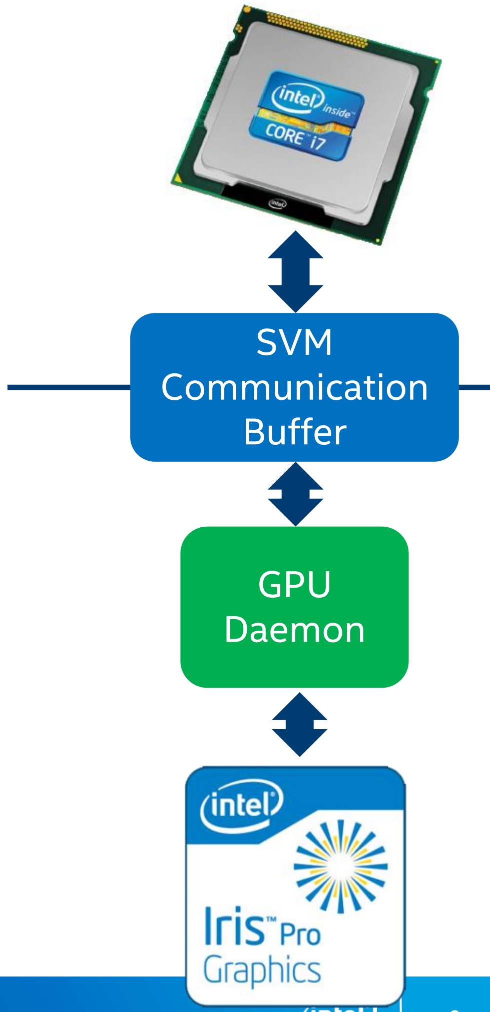
Agenda

- Current OpenCL™ scheduling model
- <u>GPU Daemon</u>:
  - Instant Mode
  - Enqueue Mode
- Performance Data
- Efficient use of GPU Daemon patterns
- Summary

Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

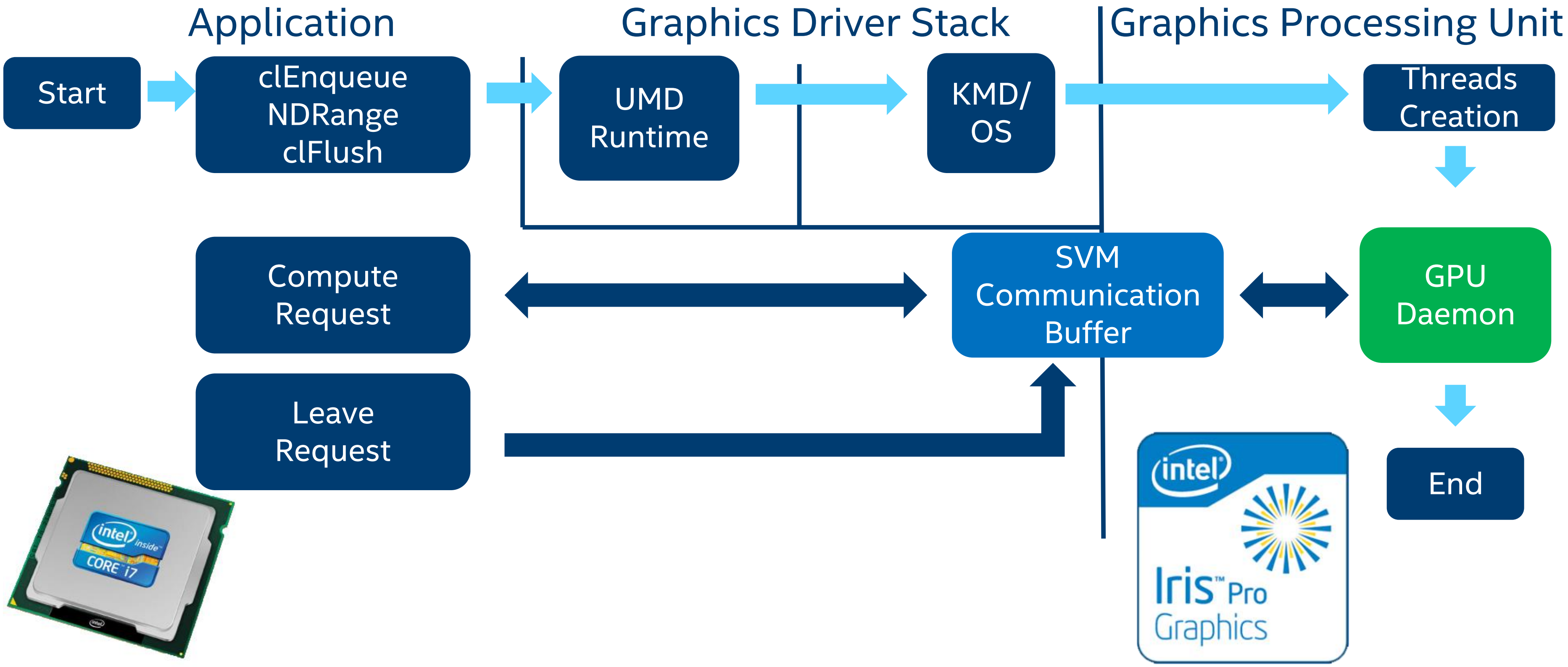
Introducing GPU Daemon

- GPU Daemon is a kernel launched from the host and later persistent on the GPU.
- It communicates with CPU using Fine-Grained Shared Virtual Memory with atomics.
- Persistency is achieved using various methods:
  - Instant Mode – loop within a kernel.
  - Enqueue Mode – self-enqueue utilizing device_enqueue.
- CPU communicates directly with active GPU threads bypassing driver stack.
- Whenever Daemon is no longer needed CPU sends “end” signal that will terminate GPU threads.



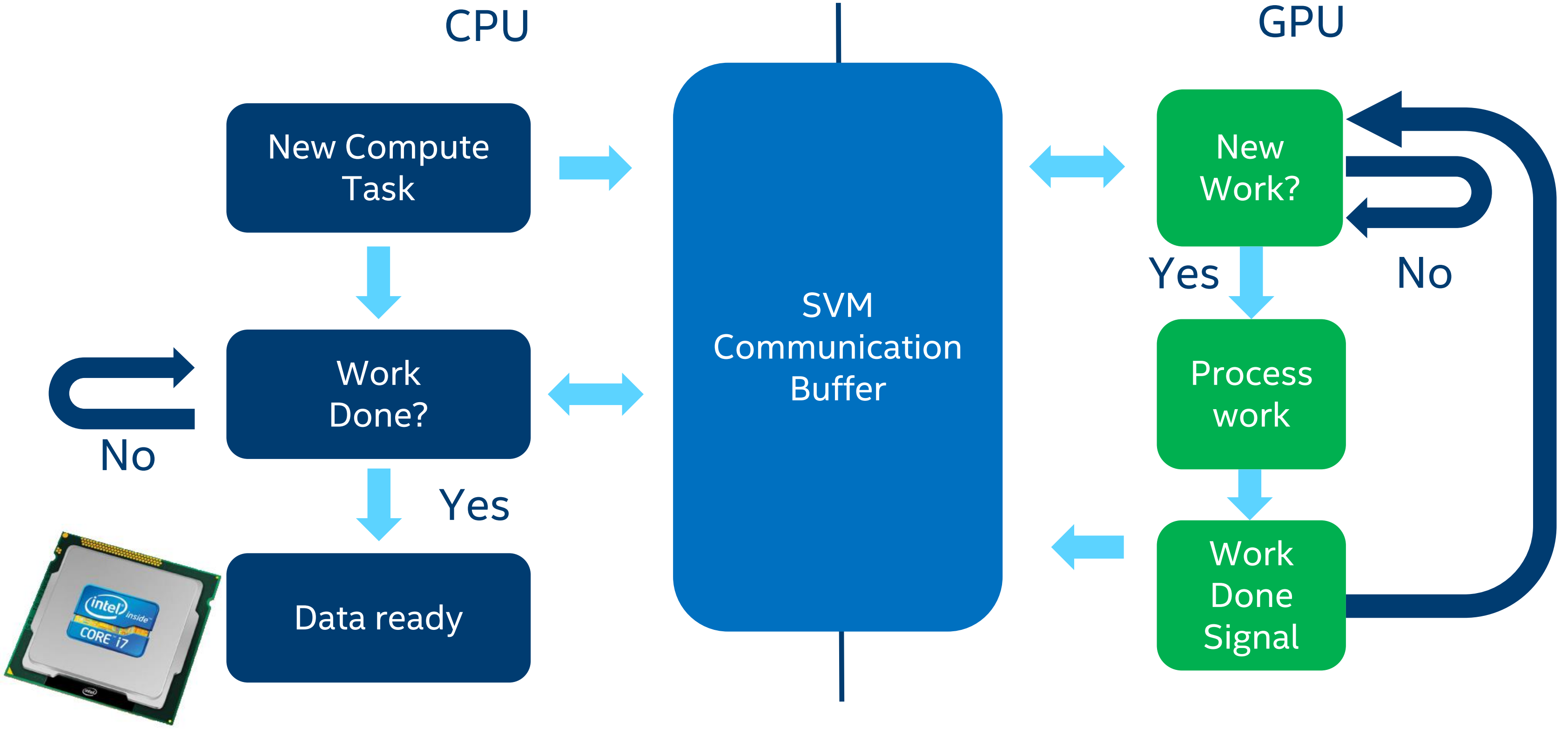
Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Introducing GPU Daemon – Instant Mode



Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

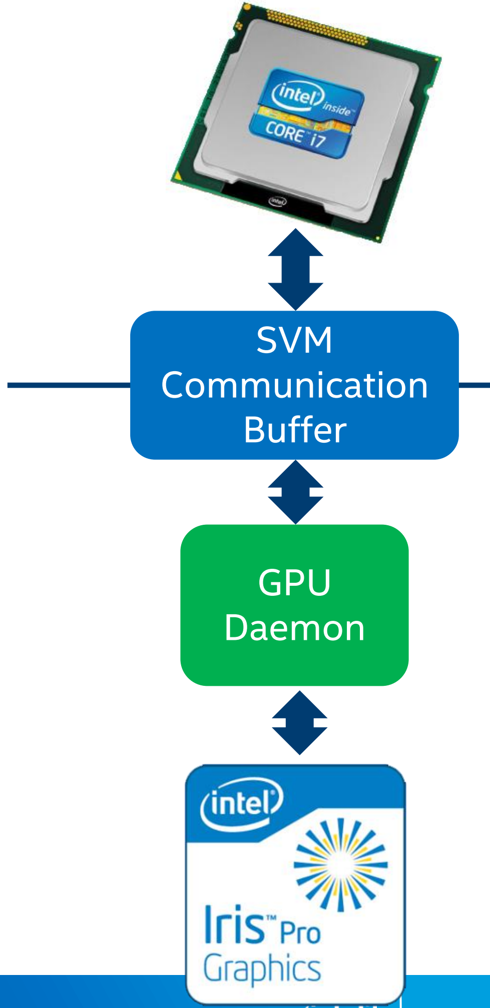
Instant Mode – tasks processing



Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

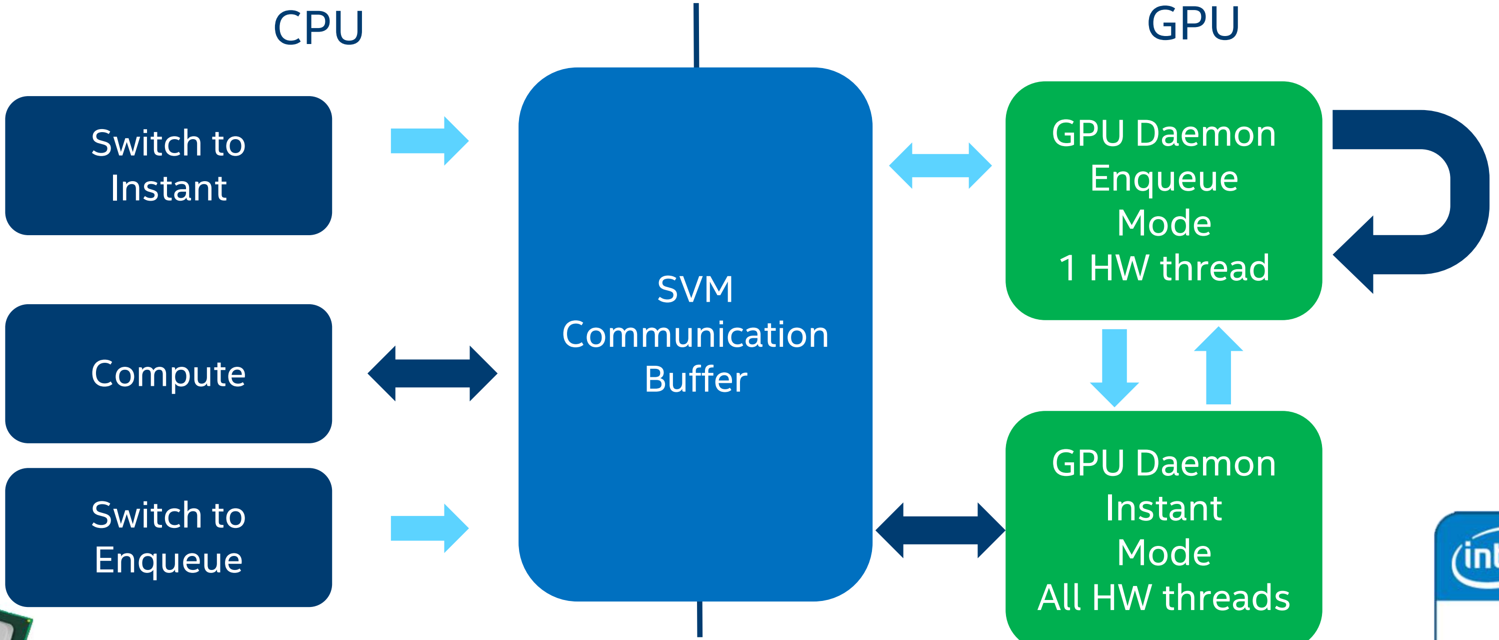
GPU Daemon in Enqueue mode

- Enqueue Mode allows various different transitions:
  - Utilizes device self-enqueue feature of OpenCL™ 2.0.
  - GPU can switch to Instant mode for direct submission.
  - GPU can enqueue traditional kernels without the need of host API interaction.
- Gives great flexibility in terms of possible options:
  - Whole host code can be transferred to the device.
  - Various Instant kernels may be dispatched, serving different compute algorithms.



Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Introducing GPU Daemon – Enqueue mode



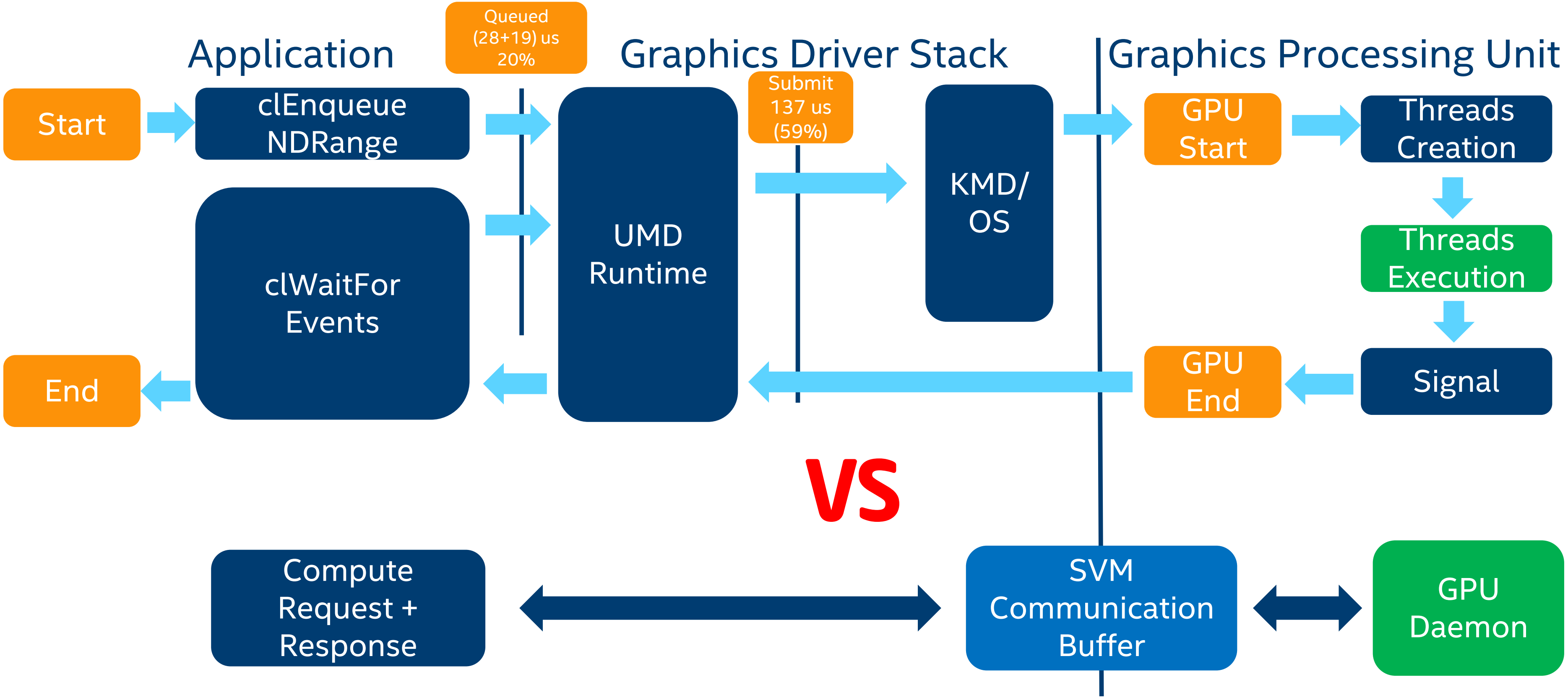
Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Agenda

- Current OpenCL™ scheduling model
- GPU Daemon:
  - Instant Mode
  - Enqueue Mode
- Performance Data
- Efficient use of GPU Daemon patterns
- Summary

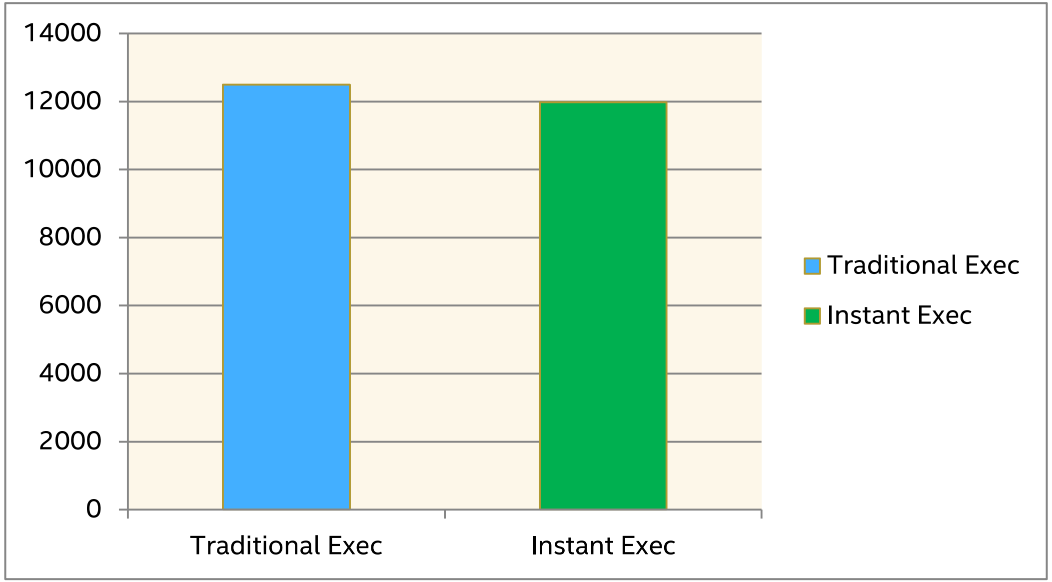
Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Model Comparison – classic vs GPU Daemon



Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Kernel execution comparison (ns)

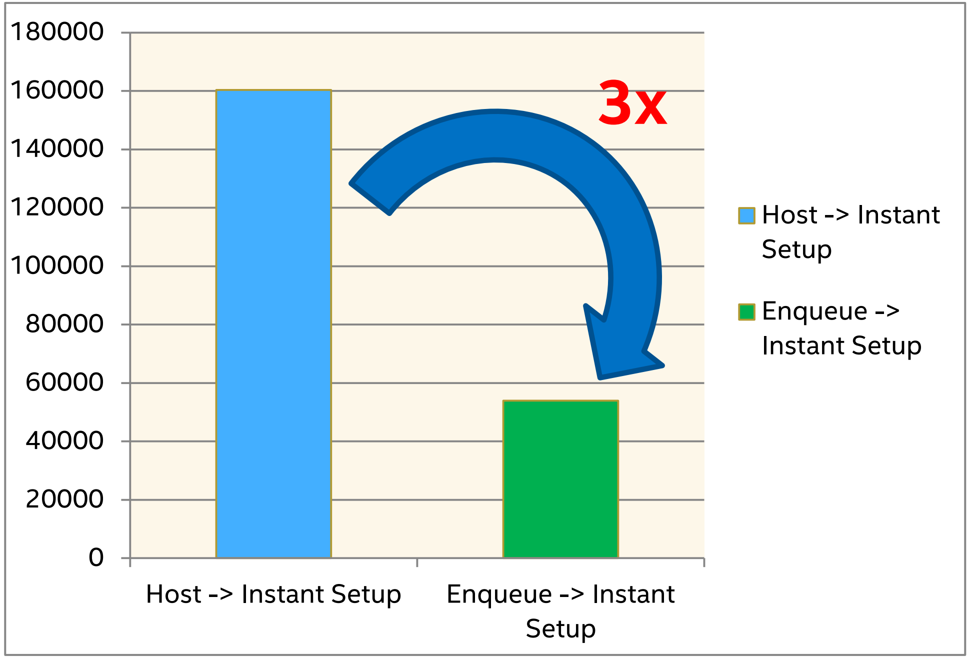


- Instant Mode Execution is faster than traditional enqueue (+5%):
  - No Thread Creation
  - No Thread Destruction
  - GPU boosted to high frequency
- This time includes CPU + GPU atomics communication cost for submission and completion.
- After work is done, threads are immediately ready for next submission.

No kernel execution overhead with CPU+GPU synchronization.

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

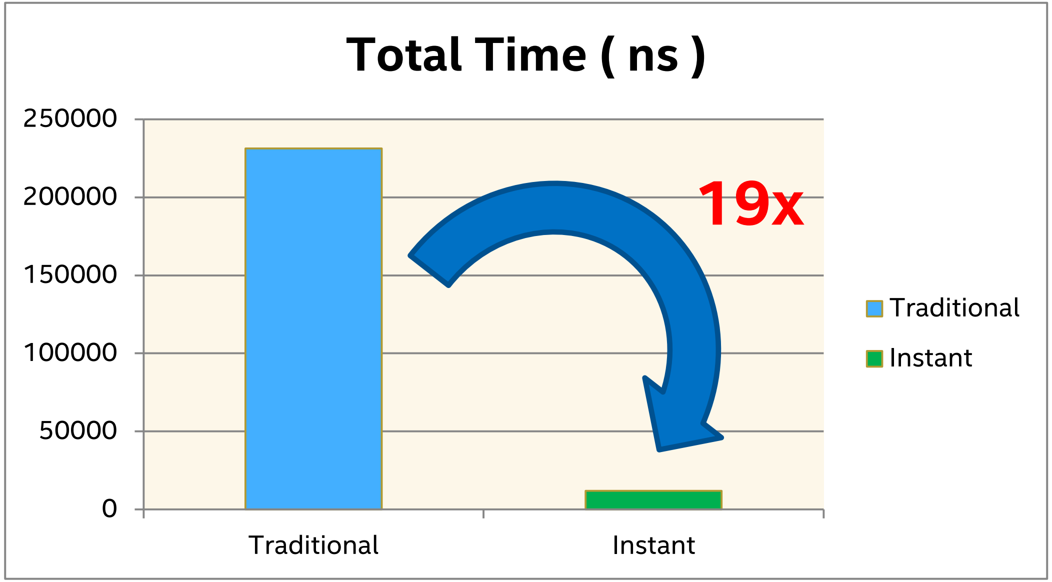
Mode Transition Latency (ns)



- Instant mode may be initiated from the host or from GPU Daemon in Enqueue Mode.
- Time needed to enter Instant Mode from the host is 160 us.
- Same operation from GPU Daemon being in Enqueue mode takes 58 us.
- Useful when multiple different instances of Instant kernels will be required.

Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Model Comparison – Instant with active Daemon



- Time from start to completion of the compute task reduced **19 times** !
- This includes submission, processing and completion of compute tasks.
- All latencies are not present, immediate compute power available on demand.

GPU Daemon is a very efficient technique for zero cost submission & completion.

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Agenda

- Current OpenCL™ scheduling model
- GPU Daemon:
  - Instant Mode
  - Enqueue Mode
- Performance Data
- Efficient use of GPU Daemon patterns
- Summary

Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Make sure you spawn all HW threads available

- Query Number of Execution Units using:

clGetDeviceInfo + CL_DEVICE_MAX_COMPUTE_UNITS

- Multiply it by number of hardware threads on each EU (typically 7, refer to device documents), this will give you total HW threads count, i.e. for Intel(R) HD Graphics 560:

24 * 7 = 168 Hardware Threads

- Obtain SIMD size of your kernel using (8,16,32):

clGetKernelWorkGroupInfo + CL_KERNEL_PREFERRED_WORK_GROUP_SIZE_MULTIPLE

- Compute global work size that will result in all threads being spawned:

Gws[0] = SIMDsize * NumberOfHwThreads = 32 * 168 = 5376

- Make sure your LWS is a multiple of SIMDsize.

- Make sure your GWS is a multiple of LWS.

Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Play nicely with GPU

- Be cautious to not spawn more HW threads than device has.
- Choose Local Work Group Size that fits nicely into sub-slices:
  - Get familiar with https://software.intel.com/sites/default/files/managed/c5/9a/The-Compute-Architecture-of-Intel-Processor-Graphics-Gen9-v1d0.pdf
  - Make sure number of HW threads per sub-slice is a multiple of HW threads per wkg.
- When using SLM(Shared Local Memory) / barriers choose bigger workgroup sizes to maximize SLM re-use:
  - Take into consideration that SLM is limited, so GPU may not spawn threads because of lack of free resources.
  - There is 64 KB per sub-slice for all workgroups, so if each uses 16KB then only 4 may be executed concurrently on this sub-slice.
- When Daemon is not needed terminate it to save power.

Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Be cautious with the amount of atomic operations

1) DON'T increment spin on every work-item:

```
__kernel Worker(__global int* pCommBuffer)
{
 __global atomic_int *atomicCommBuffer =
(__global volatile atomic_int*)pCommBuffer;

    atomic_fetch_add_explicit(
        &atomicCommBuffer[SPIN],
        1,
        memory_order_seq_cst,
        memory_scope_all_svm_devices );
    //do the work
}
```

1) DO Only single increment per thread

```
__kernel Worker(__global int* pCommBuffer)
{
    __global atomic_int *atomicCommBuffer =
(__global volatile atomic_int*)pCommBuffer;

    if( get_sub_group_local_id() == 0 )
    {
        atomic_fetch_add_explicit(
            &atomicCommBuffer[SPIN],
            1,
            memory_order_seq_cst,
            memory_scope_all_svm_devices );
     }
     //do the work
}
```

Implicit SIMD synchronization reduces the amount of atomics up to 32x.

Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Or even better, synchronize on Workgroup basis

```
__private int Finish = 0;
__private int ReqPhase = 0;
//loop as long as you need to
while( Finish != 0 )
{
    //each work item needs to check for
    work
    ReqPhase = atomic_load_explicit(
    &atomicCommBuffer[PHASE],
    memory_order_seq_cst,
     memory_scope_all_svm_devices );
    //each work item needs to obtain
    flag
    Finish = atomic_load_explicit(
    &atomicCommBuffer[FINISH],
    memory_order_seq_cst,
     memory_scope_all_svm_devices);
    //do some work
}
```

```
//shared local memory keeps control variables
__local uint Finish;
__local uint ReqPhase;
ReqPhase = Finish = 0;
barrier( CLK_LOCAL_MEM_FENCE );
//setup done, now loop as long as you need to
while(1){
    //one work item checks for completion OR new work
    if(get_local_id(0) == 0 ) {
     ReqPhase= atomic_load_explicit( &SVMComm[PHASE],
     memory_order_seq_cst,memory_scope_all_svm_devices);
     Finish= atomic_load(&SVMComm[FINISH],
     memory_order_seq_cst,memory_scope_all_svm_devices);
     //obtain work info here and propagate to SLM
    }
    barrier( CLK_LOCAL_MEM_FENCE );
    //all work items are synchronized here
    if( Finish != 0 ) return;
    //do the work on all work items basing on SLM inputs
```

Atomic traffic reduced by the factor of workgroup size (up to 256x)

*Other names and brands may be claimed as the property of others.

GPU Daemon Instant mode – Thread Spawn

CPU

```
//SVM communication buffer
std::atomic<unsigned int>*pCommBuffer =
(std::atomic<unsigned int>*)pData;
size_t gws = m_NumberOfHWThreads *
kernelSIMD;
//use 4 HW threads per WKG to minimize
atomic traffic
size_t HWThreadsPerWKG = 4;
size_t lws = kernelSIMD * HWThreadsPerWKG;

clEnqueueNDRange("InstantKernel",gws,lws);
clFlush();

//wait before GPU is ready , each thread
will signal
while(pCommBuffer[SPIN] <
m_NumberOfHWThreads);
//if we are here it means that GPU is ready
for submissions on all HW threads
```

GPU

```
__kernel InstantKernel(global int* pCommBuffer){
//tell compiler we will need atomic operations
global atomic_int *SVMComm = (global volatile
atomic_int*)pCommBuffer;
//each HW thread notifies that it is ready
if(get_sub_group_local_id() == 0) {
    atomic_fetch_add_explicit(
        &SVMComm [SPIN],1,
        memory_order_seq_cst,
        memory_scope_all_svm_devices );
}
//initialize SLM to use it later for workgroup
communication
local int Finish;
local int ReqPhase;
ReqPhase = Finish = 0;
barrier( CLK_LOCAL_MEM_FENCE );
//setup done, we may enter polling mode
```

Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

GPU Daemon Instant mode – communication

CPU

```
//make sure atomics are used
std::atomic<unsigned int>*pCommBuffer =
(std::atomic<unsigned int>*)pData;

for(uint i = 0; i < iterations; i++) {
    //triger workload
    pCommBuffer[PHASE]= ++Phase;
    //wait before completion
    while(pCommBuffer[COMPLETE]
                < m_numWorkgroups);
    //data is ready GPU completed

    //re-init completion value for next iter
    pCommBuffer[COMPLETE] = 0;
}

//terminate Instant
pCommBuffer[FINISH] = 1;
```

GPU

```
uint Phase = 0;
while(1) {
if(get_local_id(0) == 0) {
    Finish = atomic_load(&SVMComm[FINISH]);
    ReqPhase = atomic_load(&SVMComm[PHASE]);
}
barrier( CLK_LOCAL_MEM_FENCE );
if(Finish != 0) return;
if(Phase < ReqPhase) {
    //do some work, increment Phase
    Phase++;
    //now signal completion
    barrier( CLK_GLOBAL_MEM_FENCE );
    if(get_local_id(0) == 0 ) {
        atomic_fetch_add_explicit(
        &SVMComm[COMPLETE],
        1, memory_order_seq_cst,
        memory_scope_all_svm_devices );}
}
```

Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Agenda

- Current OpenCL™ scheduling model
- GPU Daemon:
  - Instant Mode
  - Enqueue Mode
- Performance Data
- Efficient use of GPU Daemon patterns
- Summary

Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Summary

- GPU Daemon is a **very** efficient technique for direct submission.
  - Submission and completion driver overhead is eliminated.
  - Kernel execution is boosted.
- GPU Daemon offers various modes allowing very flexible application paradigms
  - Instant Mode for direct submission
  - Enqueue Mode whenever we need to switch between modes or enqueue other workloads that don't require direct submission
- Get familiar with https://software.intel.com/sites/default/files/managed/c5/9a/The-Compute-Architecture-of-Intel-Processor-Graphics-Gen9-v1d0.pdf

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

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 product when combined with other products.

Copyright © 2016, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo 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

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

intel®