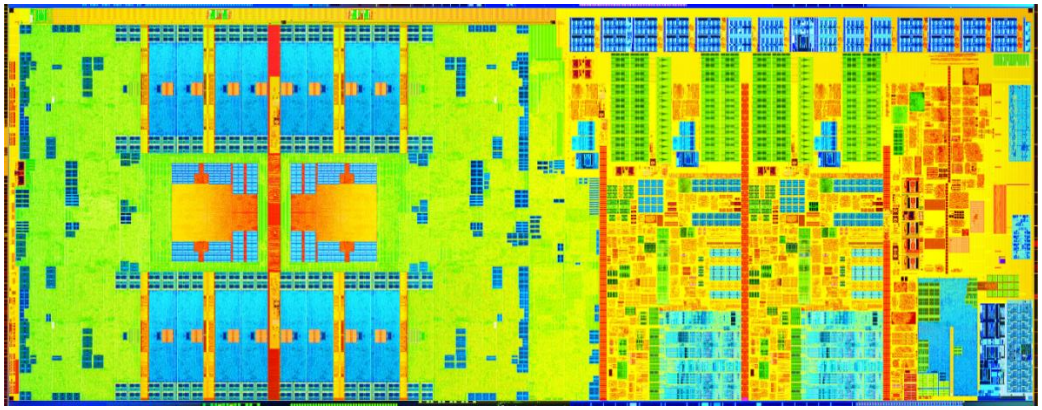
# The Compute Architecture of Intel® Processor Graphics Gen7.5

Version 1.0



External Revision History

| Version, Date | Comment |
|---|---|
| v1.0 8/1/2014 | Initial release of “The Compute Architecture of Intel® Processor Graphics Gen7.5” – Stephen Junkins |
| | |

# 1 CONTENTS

# 2 AUDIENCE

Software, hardware, and product engineers that seek to understand Intel® Processor Graphics Gen7.5's architecture. More specifically, the architecture characteristics relevant to running compute applications on Intel® Processor Graphics.

# 3 INTRODUCTION

Intel's on-die integrated Processor Graphics Architecture offers outstanding real time 3D rendering and media performance. However, its underlying compute architecture also offers general purpose compute capabilities that approach teraFLOPS performance. The architecture of Intel® Processor Graphics delivers a full complement of high throughput floating point and integer compute capabilities, a layered high bandwidth memory hierarchy, and deep integration with on-die CPUs and other on-die system-on-a-chip (**SoC**) devices. Moreover, it is a modular architecture that achieves scalability for a family of products that range from cellphones and wearables, to tablets and laptops, to high end desktops and servers.

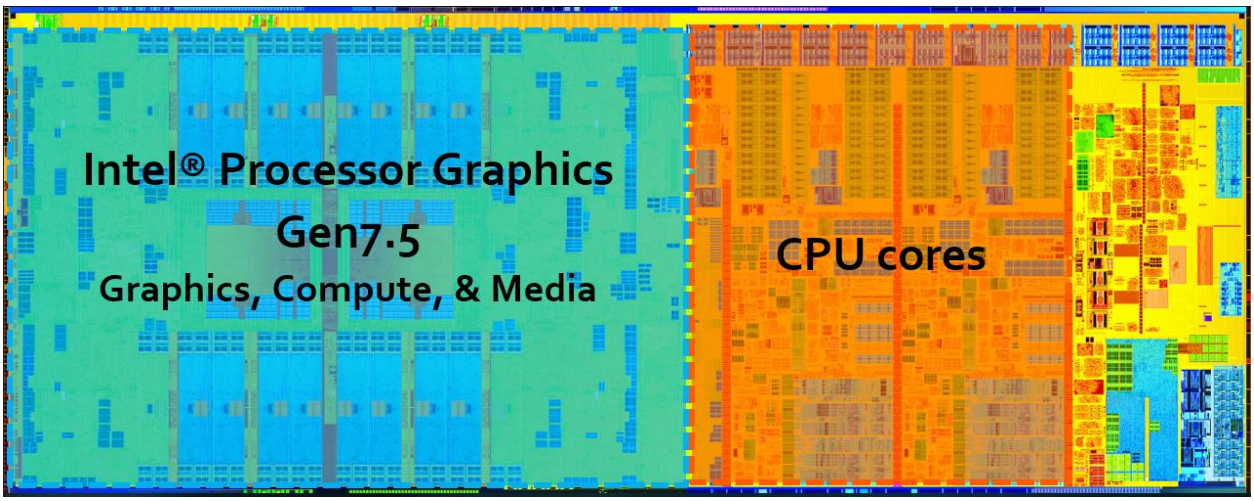



*Figure 1: Silicon die layout for 4th Generation Intel® Core™ Processor SoC. This SoC contains 2 CPU cores, outlined in orange dashed box. The on die Intel® Processor Graphics is outlined in blue dashed box.*

## 3.1 WHAT IS INTEL® PROCESSOR GRAPHICS?

**Intel® Processor Graphics** refers to the technology that provides graphics, compute, media, and display capability for many of Intel's SoC products. Within Intel, architects colloquially refer to Intel® Processor Graphics architecture as simply "**Gen**", short for Generation. A specific generation of the Intel® Processor Graphics architecture may be referred to as "Gen6" for generation 6, or "Gen7" for generation 7, etc. The branded products Intel® HD Graphics 4600, Intel® Iris™ Graphics 5100, and Intel® Iris™ Pro Graphics 5200 are all derived from instances of Intel® Processor Graphics Gen7.5 architecture. This whitepaper focuses on just the compute architecture within Intel Processor Graphics Gen7.5. For shorthand, in this paper we may use the term **Gen7.5 compute architecture** to refer to just those compute components. These compute components interoperate with other Intel® Processor Graphics components that support graphics, media, and display, but those are not discussed here. The whitepaper also

briefly discusses the instantiation of Intel Processor Graphics Gen7.5 within the 4th Generation Intel® Core™ Processor Family products.

# 4 SoC Architecture

This section describes the SoC architecture within which Intel® Processor Graphics is a component.

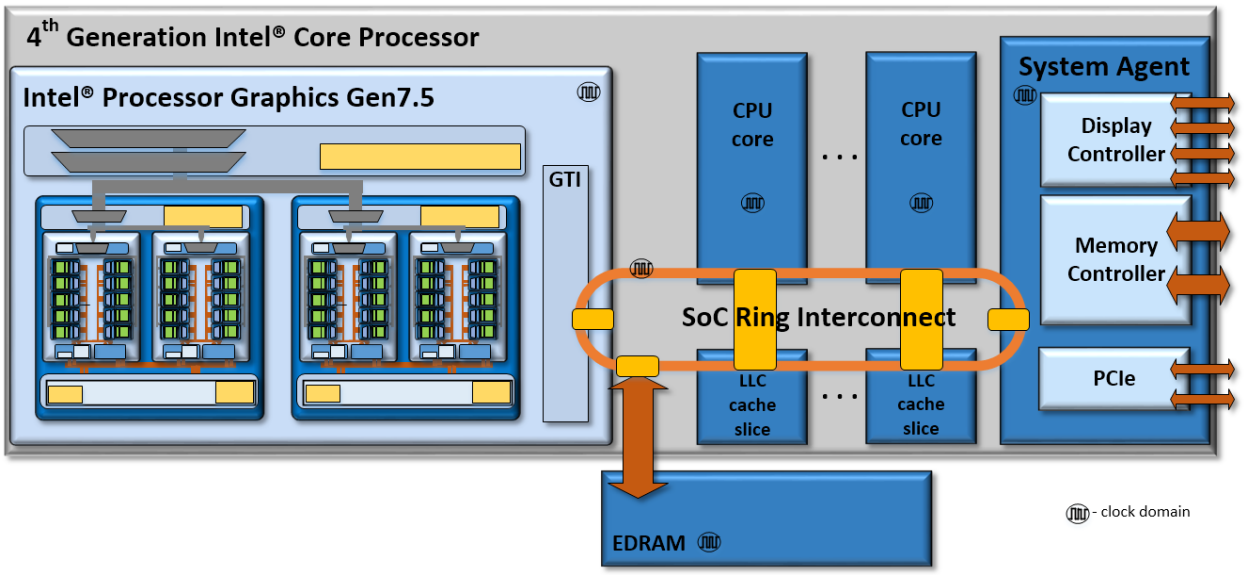



*Figure 2: 4th Generation Intel® Core™ Processor SoC and its ring interconnect architecture.*

## 4.1 SoC Architecture

The 4th Generation Intel® Core™ Processor family of microprocessors are complex SoCs integrating multiple CPU Cores, Intel® Processor Graphics, and potentially other fixed functions all on a single shared silicon die. The architecture implements multiple unique clock domains, which have been partitioned as a per-CPU Core clock domain, a Processor Graphics clock domain, and a ring interconnect clock domain. The SoC architecture is designed to be highly extensible for a range of products, and yet still enable efficient wire routing between components within the SoC.

## 4.2 Ring Interconnect

The on die bus between CPU cores, caches, and Intel® Processor Graphics is a ring based interconnect with an interface stop local to each "agent" connected to the ring. This **ring interconnect** is a bi-directional ring that has a 32-byte wide data bus, with separated lines for request, snoop, and acknowledge. Every on die CPU core is regarded as a unique agent. Similarly, Intel® Processor Graphics is also treated as a unique agent on the interconnect ring. A **system agent** is also connected to the ring, which bundles the DRAM memory management unit and display controller. Thus all communication between cores or between Intel® Processor Graphics or to/from system memory is facilitated by this interconnect and thru the system agent.

### 4.3 SHARED LLC

Some SoC products include a shared **Last Level Cache** (**LLC**) which is also connected to the ring. In such SoCs, each on die core is allocated a slice of cache, and that cache slice is connected as a unique agent on the ring. However all of the slices work together as a single cache, albeit a shared distributed cache. An address hashing scheme routes data requests to the cache slice in which the data lives, if cached. This distributed LLC is also shared with Inte® Processor Graphics. For both CPU Cores and for Intel® Processor Graphics, LLC seeks to reduce apparent latency to system DRAM and to provide higher effective bandwidth.

### 4.4 OPTIONAL EDRAM

Some SoC products may include embedded DRAM (**EDRAM**), bundled into the SoC's chip packaging. For example, the 4th Generation Intel® Core™ Processor SoC with Intel® Iris™ Pro 5200 bundles a 128 megabyte EDRAM. The EDRAM operates in its own clock domain and can be clocked up to 1.6 GHz. The EDRAM has separate buses for read and write, and each are capable of 32 byte/EDRAM-cycle. EDRAM supports many applications including low latency display surface refresh. For the compute architecture of Intel® Processor Graphics Gen7.5, EDRAM further supports the memory hierarchy by serving as a large "victim cache" behind LLC. Compute data first populates the LLC. Cacheline victims that are evicted from LLC, will spill into the EDRAM. The compute architecture of Intel® Processor Graphics can then read and write directly to or from EDRAM.

## 5 THE COMPUTE ARCHITECTURE OF INTEL® PROCESSOR GRAPHICS GEN7.5

### 5.1 MODULAR DESIGN FOR PRODUCT SCALABILITY

The Gen7.5 compute architecture is designed for scalability across a wide range of target products. The architecture's modularity enables exact product targeting to a particular market segment or product power envelope. The architecture begins with compute components called execution units. Execution units are clustered into groups called subslices. Subslices are further clustered into slices. Together, execution units, subslices, and slices are the modular building blocks that are composed to create many product variants based upon Gen7.5 compute architecture. The following sections describe the architecture components in detail, and show holistically how they may be composed into full products.

### 5.2 EXECUTION UNIT (EUS) ARCHITECTURE

The foundational building block of Gen7.5 compute architecture is the **execution unit**, commonly abbreviated as just **EU**. EUs are Simultaneous Multi-Threading (**SMT**) compute processors that drive multiple issue Single Instruction Multiple Data Arithmetic Logic Units (**SIMD**, **ALU**) pipelined for high throughput floating point and integer compute. The highly threaded nature of the EUs ensures continuous streams of ready to execute instructions, while also enabling latency hiding of longer operations such as memory scatter/gather, sampler requests, or other system communications.

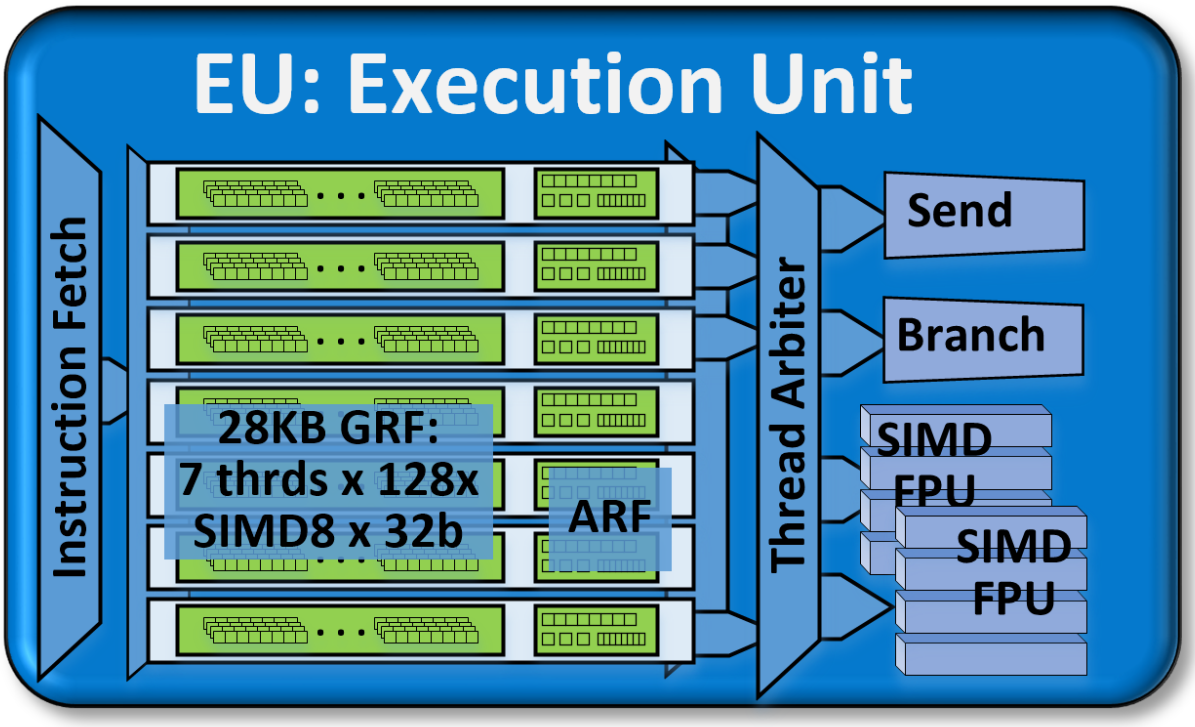



*Figure 3: The Execution Unit (EU). Each Gen7.5 EU has seven threads. Each thread has 128 SIMD-8 32-bit registers (GRF) and supporting architecture specific registers (ARF). The EU can co-issue to four instruction processing units including two FPUs, a branch unit, and a message send unit.*

Product architects may fine tune the number of threads and number of registers per EU to match scalability and specific product design requirements. For Gen7.5 based products, each EU thread has 128 general purpose registers. Each register stores 32 bytes, accessible as a SIMD 8 element vector of 32-bit data elements. For Gen7.5 based products, each EU has seven threads for a total of 28 Kbytes of general purpose register file (**GRF**). Flexible addressing modes permit registers to be addressed together to build effectively wider registers, or even to represent strided rectangular block data structures. Per thread architectural state is maintained in a separate dedicated architecture register file (**ARF**).

### 5.2.1 Simultaneous Multi-Threading and Multiple Issue Execution

Depending on the software workload, the hardware threads within an EU may be executing the same compute kernel code, or each EU thread could be executing completely different compute kernel code. The execution state of each thread, including its own instruction pointer stack, are held in a thread specific ARF registers.

On every cycle, an EU can co-issue up to four different instructions, from four different threads. The EU's **thread arbiter** dispatches these instructions to one of four issue slots for execution. Although the issue slots pose some instruction co-issue constraints, the four instructions will be independent, since they are dispatched from four separate threads.

Within the EUs, branch instructions are dispatched to a dedicated **branch unit** to facilitate SIMD divergence and eventual convergence. Finally, memory operations, sampler operations, and other system communications are all dispatched via a message passing **send unit**.

### 5.2.2 SIMD FPUs

Within each EU, the primary computation units are a pair of SIMD floating point units (**FPU**). Although called FPUs, they support both floating point and integer computation. These units can SIMD execute up to four 32-bit floating point (or integer) operations, or SIMD execute up to eight 16-bit integer operations. Each SIMD FPU can complete simultaneous add and multiply (MAD) floating point instructions every cycle. Thus each EU is capable of 16 32-bit floating point operations per cycle: (add + mul) x 2 FPUs x SIMD-4. One of the FPUs also supports high throughput SIMD integer operations. Finally, one FPU includes extended math capability and supports high throughput transcendental math functions. For Intel® Iris™ Pro 5200 based products, the theoretical peak aggregate FLOPS across the entire architecture can approach 832 GFLOPS.

Gen7.5 compute architecture offers significant local bandwidth within each EU between GRF and the FPUs. For example, MAD style instructions with three source operands and one destination operand are capable of driving 96 bytes/cycle read bandwidth, and 32 bytes/cycle write bandwidth locally within every EU. Aggregated across the whole architecture, this bandwidth can scale linearly with the number of EUs. For Intel® Iris™ Pro 5200 based products, the aggregated theoretical peak bandwidth that is local between FPUs and GRF can approach multiple terabytes of read bandwidth.

### 5.2.3 EU ISA and Flexible Width SIMD

The EU Instruction Set Architecture (**ISA**) and supporting general purpose register file are all designed to support a flexible SIMD width. Thus for 32-bit data types, the Gen7.5 FPUs can be viewed as *physically* 4-wide. But the FPUs may be targeted with SIMD instructions and registers that are *logically* 1-wide, 2-wide, 4-wide, 8-wide, 16-wide, or 32-wide. For example, a single operand to a SIMD-16 wide instruction pairs two adjacent SIMD-8 wide registers, addressing the pair as a logical single SIMD-16 wide register containing a contiguous 64 bytes.

Wider SIMD instructions take more cycles to complete execution, but there is no performance restriction for mixing SIMD instruction-widths. The instruction width choice is left to the compiler or low level programmer. Differing SIMD width instructions can be issued back to back. This flexible design allows compilers and programmers to choose specific width of SIMD to precisely manage register allocation footprint for individual programs, balanced against the amount of work assigned to each thread.

Compilers for Single Program Multiple Data (**SPMD**) programming models such as Renderscript, OpenCL™[1], Microsoft® DirectX®[2] Compute Shader, and C++AMP, generate SIMD code to map multiple **kernel instances**[3] to be executed simultaneously within a given hardware thread. The exact number of kernel instances per thread is a heuristic driven compiler choice. We refer to this compiler choice as the dominant **SIMD-width** of the kernel. In OpenCL and DirectX Compute Shader, SIMD-8, SIMD-16, SIMD-32 are common SIMD-width examples.

On Gen7.5 compute architecture, most SPMD programming models employ this style code generation and EU processor execution. Effectively, each SPMD kernel instance appears to

[1] *OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.*
[2] Microsoft and DirectX are either registered trademarks or trademarks of Microsoft Corporation in the United Stated and/or other countries.
[3] We use the generic term *kernel instance* as equivalent to OpenCL *work-item*, or DirectX Compute Shader *thread*.

execute serially and independently within its own SIMD lane. In actuality, each thread executes a SIMD-Width number of kernel instances concurrently. Thus for a SIMD-16 compile of a compute kernel, it is possible for SIMD-16 x 7 threads = 112 kernel instances to be executing concurrently on a single EU. Similarly, for SIMD-32 x 7 threads = 224 kernel instances executing concurrently on a single EU.

For a given SIMD-width, if all kernel instances within a thread are executing the same instruction, then the SIMD lanes can be maximally utilized. If one or more of the kernel instances chooses a divergent branch, then the thread will execute the two paths of the branch separately in serial. The EUs branch unit keeps track of such branch divergence and branch nesting. The branch unit also generates a "liveness" mask to indicate which kernel instances within the current SIMD-width need to execute (or not execute) the branch.

## 5.3 Subslice Architecture

In Gen7.5 compute architecture, arrays of EUs are instantiated in a group called a **subslice**. For scalability, product architects have choice as to the exact number of EUs per subslice. For most Gen7.5 based products, each subslice contains 10 EUs. Each subslice contains its own **local thread dispatcher** unit and its own supporting instruction caches. Given these 10 EUs with 7 threads each, a single subslice has dedicated hardware resources and register files for a total of 70 simultaneous threads. Each subslice also includes a sampler unit and a data port memory management unit.

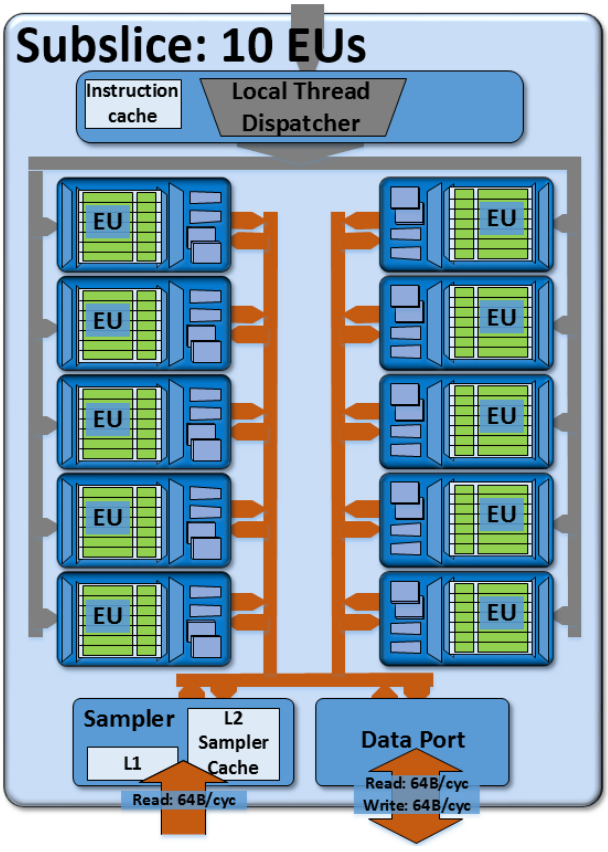



*Figure 4: The Intel® Processor Graphics Gen7.5 subslice, containing 10 EUs each. The subslice also instantiates sampler and data port units per subslice.*

### 5.3.1 Sampler

The **sampler** is a read only memory fetch unit that may be used for sampling of tiled (or not tiled) texture and image surfaces. The sampler includes a level-1 sampler cache (**L1**) and a level-2 sampler cache (**L2**). Between the two caches is dedicated logic to support dynamic decompression of block compression texture formats such as DirectX BC1-BC7, DXT, and OpenGL compressed texture formats. The sampler also includes fixed function logic that enables address conversion on image (u,v) coordinates, and address clamping modes such as mirror, wrap, border, and clamp. Finally the sampler supports a variety of sampling filtering modes such as point, bilinear, tri-linear, and anisotropic.

### 5.3.2 Data Port

Each subslice also contains a memory load/store unit called the **data port**. The data port supports efficient read/write operations for a variety of general purpose buffer accesses, flexible SIMD scatter/gather operations, as well as shared local memory access. To maximize memory bandwidth, the unit also dynamically coalesces scattered memory operations into fewer operations over non-duplicated 64-byte cache line requests. For example, a SIMD-16 gather operation against 16 unique offset addresses for 16 32-bit floating point values, might be coalesced to a single 64-byte read operation if all the addresses fall within a single cacheline.

## 5.4 Slice Architecture

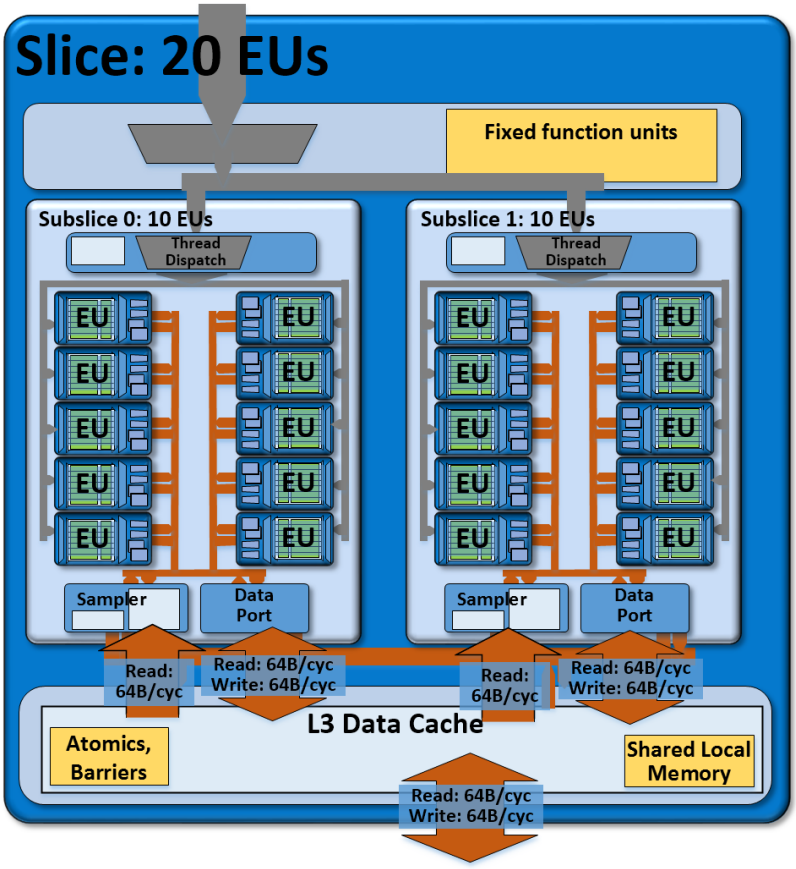



*Figure 5: The Intel® Processor Graphics Gen7.5 slice, containing two subslices for a total of 20 EUs. The slice also adds supporting L3 cache, shared local memory, atomics, barriers, and other supporting fixed function.*

Subslices are clustered into **slices**. For Gen7.5 based products, 2 subslices are aggregated into 1 slice. Thus a single slice aggregates a total of 20 EUs. Aside from grouping subslices, the slice also integrates additional logic for thread dispatch routing, a banked level-3 cache, a smaller but highly banked shared local memory structure, and fixed function logic for atomics and barriers. Additional fixed function units also support media and graphics capability, but are not discussed here.

### 5.4.1 Level-3 Data Cache

For Gen7.5 based products, the level-3 (**L3**) data cache is allocated as 256 Kbytes total per slice. Products with multiple slices, will instantiate multiple L3 cache slices. These cache slices aggregate together and act as a single larger capacity monolithic cache. Cachelines are 64 bytes and are uniformly distributed across the entire aggregate cache. Every sampler and every data port are given their own separate memory interface to the L3. The interface between each data port and the L3 data cache enables both read and write of 64 bytes per cycle. Thus a slice containing two subslices, each with a unique data port, will have an aggregate bandwidth of 128 bytes per cycle. For accesses that miss the L3 cache, the L3 fill logic can read and write system memory data at 64 bytes per cycle.

All data into and out of the samplers and data ports flows through the L3 data cache in units of 64-byte wide cachelines. This includes read and write actions on general purpose buffers. It also includes sampler read transactions that miss the level-1 (L1) and level-2 (L2) sampler caches. L3 cache bandwidth efficiency is highest for read/write accesses that are cacheline aligned and adjacent within cacheline. Compute kernel instructions that miss the subslice instruction caches also flow through the L3 cache.

### 5.4.2 Shared Local Memory

**Shared local memory**[4] is a dedicated structure within the L3 complex that supports programmer managed data for sharing amongst kernel instances within an OpenCL work-group or within a DirectX11 threadgroup. The read/write bus interface between each subslice and shared local memory is again 64-bytes wide. The shared local memory itself is more highly banked than the L3 caches. Its organization can yield full shared local memory bandwidth for access patterns that may not be 64-byte aligned or that may not be contiguously adjacent in memory. For Gen7.5 based products, 64Kbytes of shared local memory is available per subslice. SPMD programming model constructs such as OpenCL's *local* memory space or DirectX Compute Shader's *shared* memory space constrains work-group accesses to be between a single shared local memory partition and a single subslice. Because of this property, an application's accesses to shared local memory should scale with the number of subslices.

### 5.4.3 Barriers and Atomics

Each slice within Gen7.5 compute architecture also bundles dedicated logic to support implementation of group barriers. This barrier logic is available as a hardware alternative to pure compiler based barrier implementation approaches. The Gen7.5 logic can support barriers simultaneously in 16 active work-groups per subslice.

[4] We use the term *shared local memory* to indicate the memory structure that supports the memory address spaces that OpenCL refers to as *work-group local memory* and that DirectX Compute Shader refers to as *thread group shared memory*.

Each slice also provides a rich suite of atomic read-modify-write memory operations. These operations support both operations to L3 cached memory or to shared local memory. Gen7.5 based products support 32-bit atomic operations.

### 5.4.4 64-Byte Data Width

A foundational element of Gen7.5 compute architecture is the **64-byte data width**. Recall that the EU register file is composed of 128 32-byte registers (SIMD-8 x 32-bit). But recall also that operands to SIMD-16 instructions typically pair two such registers, treating the pair as a single 64-byte SIMD-16 register. Observe:

- A SIMD-16 instruction can source 64-byte wide operands from 64-byte wide registers.
- The data for such 64-byte wide registers are read and written from L3 over a 64-byte wide data bus.
- Within the L3 data cache, each cacheline is again 64-bytes wide.
- Finally the L3 cache's bus interface to the SoC shared LLC is also 64-bytes wide.

## 5.5 Product Architecture

Finally, SoC product architects can create product families or a specific product within a family by instantiating a single slice, or groups of slices. Members of a product family might differ primarily in the number of component slices. These slices are combined with additional front end logic to manage command submission, as well as fixed function logic to support 3D, rendering, and media pipelines. Additionally the entire Gen7.5 compute architecture interfaces to the rest of the SoC components via a dedicated interface unit called the Graphics Technology Interface (GTI).

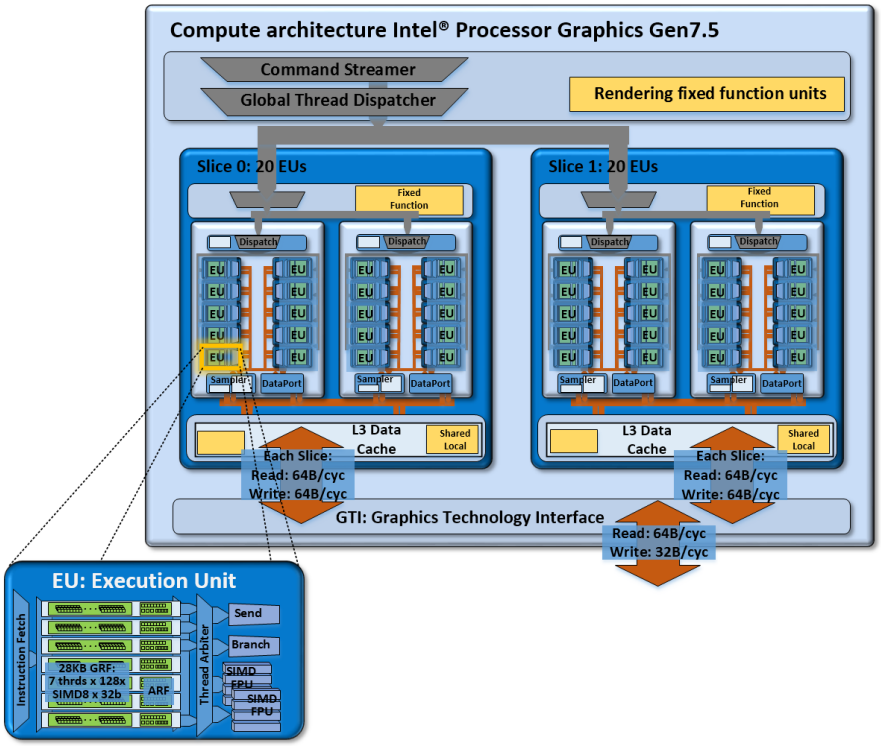



*Figure 6: A product instantiation of the compute architecture of Intel® Processor Graphics Gen7.5 for Intel® Iris™ Pro 5200. It is composed of two slices, of two subslices each for a total of 40 EUs.*

### 5.5.1 Command Streamer and Global Thread Dispatcher

As its name implies, the **command streamer** efficiently parses command streams submitted from driver stacks and routes individual commands to their representative units. For compute workloads, the **global thread dispatch** unit is responsible for load balancing thread distribution across the entire device. The global thread dispatcher works in concert with thread dispatch local logic on each subslice.

The global thread dispatch unit operates in two modes. For compute workloads that *do not depend* upon hardware barriers nor upon shared local memory, thread dispatch global may choose to distribute the workload over all available slices to maximize throughput and utilization. Given the unit's global visibility, it is able to load balance across all the execution resources. For compute workloads that *do depend* upon hardware barriers or shared local memory, thread dispatch global will assign work-group (aka thread-group) sized portions of the workload to specific subslices. Such an assignment ensures localized access to the barrier logic and shared local memory storage dedicated to each subslice.

### 5.5.2 Graphics Technology Interface (GTI)

The graphics technology interface, or simply **GTI**, is the gateway between Gen7.5 compute architecture with the rest of the SoC. The rest of the SoC includes memory hierarchy units such as the shared LLC memory, the system DRAM, and possibly embedded DRAM. GTI also facilitates communication with the CPU cores, and possibly with other fixed function devices such imaging. GTI also implements global memory atomics that may be shared between Gen7.5 compute architecture and CPU cores. Finally GTI implements power management controls for Gen7.5 compute architecture and interfaces between the GTI clock domain and the (usually different) SoC clock domains.

The bus between each slice that is interfaced to GTI is capable of 64-bytes per cycle read and 64-bytes per cycle write. Internally, GTI has memory request queues to maintain memory order and manage differing latencies on individual memory requests. For instances of Intel® Processor Graphics Gen7.5, the bus between GTI and LLC is capable of 64-bytes per cycle read and 32-bytes per cycle write. Like other aspects of the architecture, this bus width is also scalable, and SoC designers may re-configure for specific products.

## 5.6 MEMORY

### 5.6.1 Unified Memory Architecture

Intel® Processor Graphics architecture has long pioneered sharing DRAM physical memory with the CPU. This unified memory architecture offers a number of system design, power efficiency, and programmability advantages over PCIe hosted discrete memory systems.

The obvious advantage is that **shared physical memory** enables **zero copy** buffer transfers between CPUs and Gen7.5 compute architecture. Moreover, the architecture further augments the performance of this sharing with shared caches. The net effect of this architecture benefits performance, conserves memory footprint, and indirectly conserves system power not spent needlessly copying data. Shared physical memory and zero copy buffer transfers are programmable though the buffer allocation mechanisms in APIs such as OpenCL 1.0+ and DirectX11 DX11.2+.

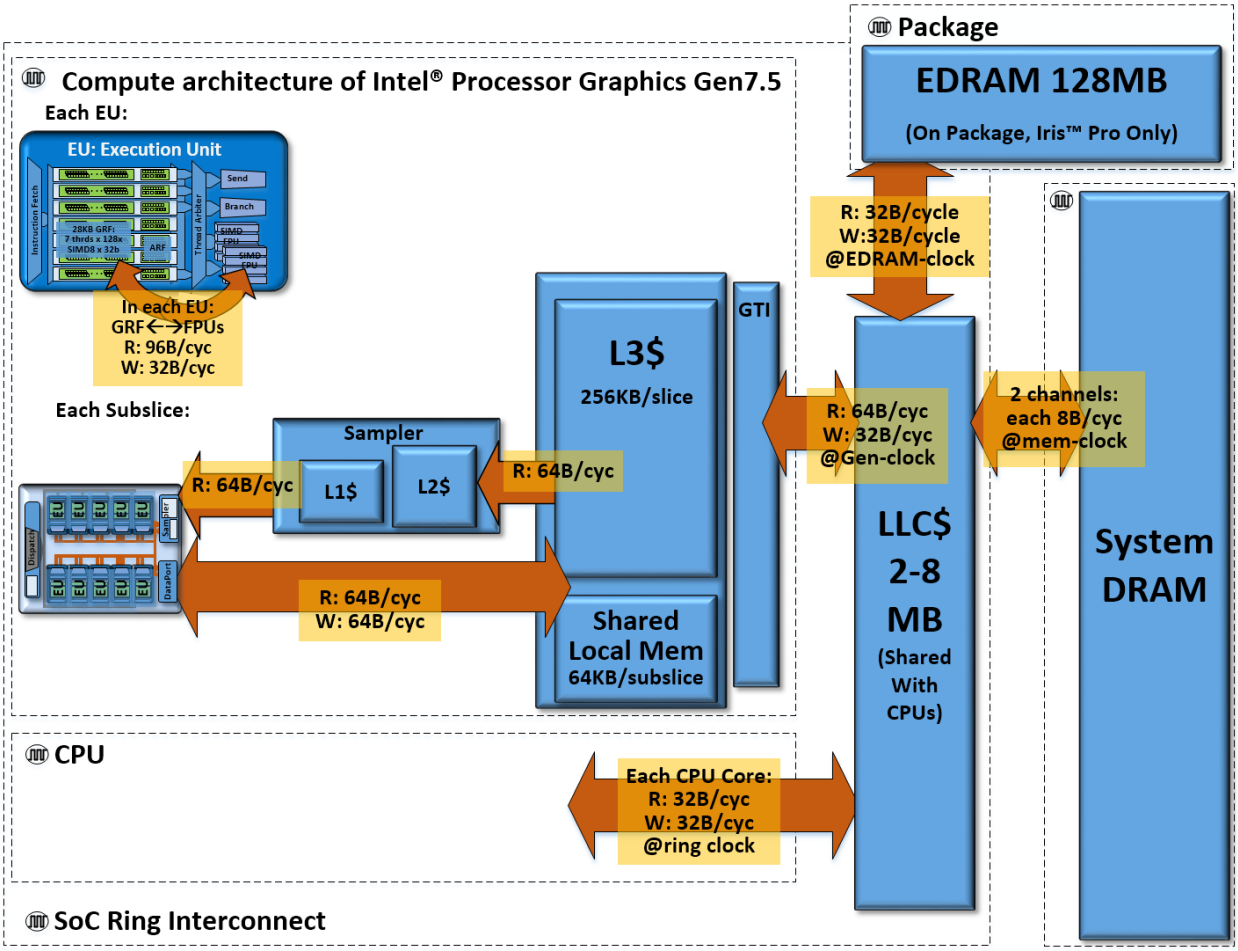



*Figure 7: A view of the memory hierarchy and its theoretical peak bandwidths for the compute architecture of Intel® Processor Graphics.*

## 5.7 Architecture Configurations, Speeds, and Feeds

The following table presents the *theoretical peak performance* of the compute architecture of Intel® Processor Graphics, *aggregated* across the entire architecture. The product max clock rate assumed in the peak performance computations is stated when necessary.

| | Intel ® HD Graphics 4600 | Intel® Iris™ Pro Graphics 5200 | Intel® Iris™ Pro Derivation, notes |
|---|---|---|---|
| **Configurations:** | | | |
| Execution Units (EUs) | 20 EUs | 40 EUs | 10 EUs x 2 subslices x 2 slices |
| Simultaneous HW Threads | 140 threads | 280 threads | 40 EUs x 7 threads |
| Concurrent kernel instances (e.g. OpenCL work-items or DirectX Compute Shader “threads”) | 4480 instances | 8960 instances | 280 threads * SIMD-32 compile |
| Level-3 data cache (L3$) | 256 Kbytes | 512 Kbytes | 2 slices x 256 Kbytes /slice |
| Max Shared Local Memory | 128 Kbytes | 256 Kbytes | 4 subslices |

| | | | |
|---|---|---|---|
| | | | x 64 Kbytes /subslice |
| On die Last Level Cache (LLC$) | 2-8 Mbytes | 2-8 Mbytes | depending on product config |
| On package embedded DRAM | n/a | 128 Mbytes | |
| **Peak Compute Throughput** | | | |
| 32b float FLOPS | 320 FLOP/cycle | 640 FLOP/cycle<br>or<br>**832 GFLOPS@1.3GHz** | 40 EUs<br>x (2 x SIMD4 FPU)<br>x (MUL + ADD) |
| 64b double float FLOPS | 80 FLOP/cycle | 160 FLOP/cycle<br>or<br>**208 GFLOPS@1.3Ghz** | 40 EUs<br>x SIMD4 FPU<br>x (MUL + ADD)<br>x ½ throughput |
| 32b integer IOPS | 80 IOP/cycle | 160 IOP/cycle<br>or<br>**208 GIOPS@1.3GHz** | 40 EUs<br>x SIMD4 FPU<br>x (ADD) |

# 6 COMPUTE APPLICATIONS

The following images provide a few visual examples of the kinds of compute applications and algorithms that have been accelerated on Intel® Processor Graphics.


*Figure 8: Intel® Processor Graphics acceleration in Adobe Photoshop, via OpenCL. In the left pair of images, Adobe Photoshop's all new "Smart Sharpen" feature uses Intel® Processor Graphics to efficiently analyze images to maximize clarity and minimize noise and halos. In the right pair of images, Adobe Photoshop's "Intelligent upsampling" features use Inte® Processor Graphics to accelerate upsampling operations that preserve detail and sharpness without introducing noise. Images courtesy of Anita Banerjee and Ilya Albrekht.*

*Figure 9: Real-time volumetric rendered 3D smoke effect implemented using DirectX11 Compute Shader on Intel® Processor Graphics. Image courtesy Doug McNabb.*

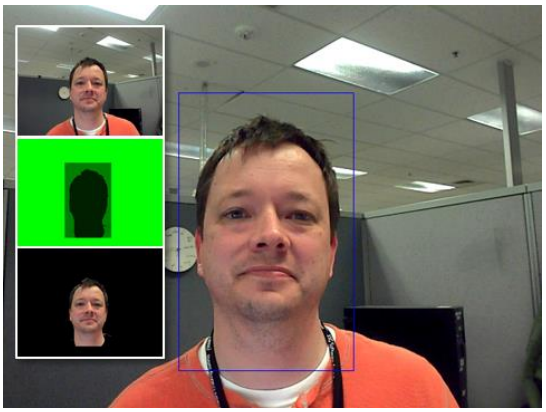


*Figure 10: OpenCV Face detection algorithm, accelerated via OpenCL on Intel® Processor Graphics. Image courtesy Aaron Kunze.*

# 7 MORE INFORMATION

- Intel® Iris™ Graphics Powers Built-in Beautiful
- About Intel® Processor Graphics Technology
- Open source Linux documentation of Gen Graphics and Compute Architecture
- Intel® OpenCL SDK
- Optimizing Heterogeneous Computing for Intel® Processor Graphics, IDF 2014 Shenzhen
- Intel® 64 and IA-32 Architectures Software Developers Manual

# 8 NOTICES

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

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.

Intel, the Intel logo, Iris™, Core™ are trademarks of Intel Corporation in the U.S. and/or other countries.

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

Intel® Graphics 4600, Iris™ Graphics, and Iris™ Pro Graphics are available on select systems. Consult your system manufacturer. visit http://www.intel.com/content/www/us/en/architecture-and-technology/microarchitecture/latest-microarchitecture.html