Intel® UHD Graphics Open Source

Programmer's Reference Manual

For the 2020 Intel Core™ Processors with Intel Hybrid Technology based on the "Lakefield" Platform

Volume 3: GPU Overview

May 2021, Revision 1.0

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks.

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described 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.

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.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

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.

© Intel Corporation. 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.

Table of Contents

Overview

The integrated graphics component, specifically called the Graphics Processing Unit, or GPU, resides on the same chip die as the Central Processing Unit, or CPU, and communicates with the CPU via the on-chip bus, with internal memory and with output device(s). As Intel GPUs have evolved, they now occupy a significant percentage of space on the chip and provide customers with high performance and low-power graphics processing, eliminating the need to purchase a separate video card for most users.

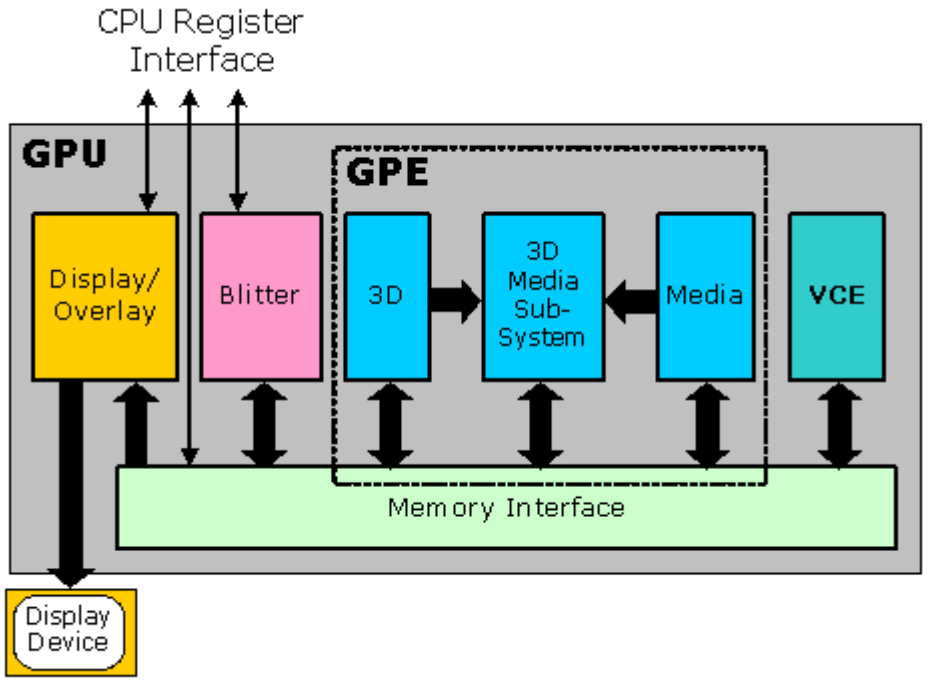
This Programmer's Reference Manual (PRM), provides detailed narrative and referential information required by graphics device driver engineers and graphics API-level programmers to take advantage of the sophisticated architecture and programmability of the GPU.

Graphics Processing Unit (GPU)

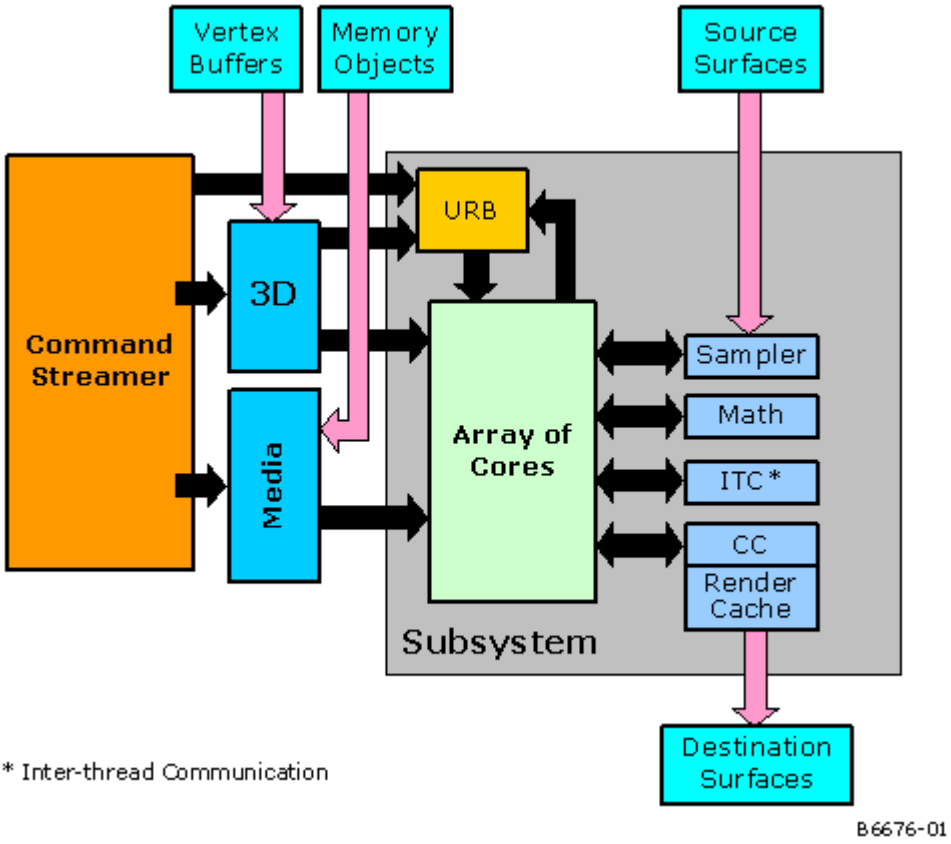
The Graphics Processing Unit is controlled by the CPU through a direct interface of memory-mapped IO registers, and indirectly by parsing commands that the CPU has placed in memory. The Display interface and Blitter (**bl**ock **i**mage **t**ransfer**er**) are controlled primarily by direct CPU register addresses, while the 3D and Media pipelines, part of Graphics Processing Engines (GPE) and the parallel Video Codec Engine (VCE) are controlled primarily through instruction lists in memory.

The subsystem contains an array of cores, or execution units, with a number of "shared functions", which receive and process messages at the request of programs running on the cores. The shared functions perform critical tasks, such as sampling textures and updating the render target (usually the frame buffer). The cores themselves are described by an instruction set architecture, or ISA.

Block Diagram of the GPU



Rendering Engine Overview



The subsystem consists of an array of *execution units* (*EUs*, sometimes referred to as an array of *cores*) along with a set of *shared functions* outside the EUs that the EUs leverage for I/O and for complex computations. Programmers access the subsystem via the 3D or Media pipelines.

EUs are general-purpose programmable cores that support a rich instruction set that has been optimized to support various 3D API shader languages as well as media functions (primarily video) processing.

Shared functions are hardware units which serve to provide specialized supplemental functionality for the EUs. A shared function is implemented where the demand for a given specialized function is insufficient to justify the costs on a per-EU basis. Instead a single instantiation of that specialized function is implemented as a stand-alone entity outside the EUs and shared among the EUs.

Invocation of the shared functionality is performed via a communication mechanism called a *message*. A message is a small self-contained packet of information created by a kernel and directed to a specific shared function. The message is defined by a range of MRF registers that hold message operands, a destination shared function ID, a function-specific encoding of the desired operation, and a destination GRF register to which any writeback response is to be directed. Messages are dispatched to the shared function under software control via the send instruction. This instruction identifies the contents of the message and the GRF register locations to direct any response.

The message construction and delivery mechanisms are general in their definition and capable of supporting a wide variety of shared functions.

Command Stream (CS) Unit

The Command Stream (CS) unit manages the use of the 3D and Media pipelines; it performs switching between pipelines and forwarding command streams to the currently active pipeline. It manages allocation of the URB and helps support the Constant URB Entry (CURBE) function.

3D Pipeline

The 3D Pipeline provides specialized 3D primitive processing functions. These functions are provided by a pipeline of "fixed function" stages (units) and threads spawned by these units. See *3D Pipeline Overview*.

Media Pipeline

The Media pipeline provides both specialized media-related processing functions and the ability to perform more general ("generic") functionality. These Media-specific functions are provided by a Video Front End (VFE) unit. A Thread Spawner (TS) unit is utilized to spawn threads requested by the VFE unit, or as required when the pipeline is used for general processing. See *Media Pipeline Overview*.

Thread Dispatching

When the 3D and Media pipelines send requests for thread initiation to the Subsystem, the thread Dispatcher receives the requests. The dispatcher performs such tasks as arbitrating between concurrent requests, assigning requested threads to hardware threads on EUs, allocating register space in each EU among multiple threads, and initializing a thread's registers with data from the fixed functions and from the URB. This operation is largely transparent to software.

Execution Units (EUs)

The Execution Units (EUs) are the programmable shader units of the Architecture. Each is a stand-alone programmable computational unit used for execution of 3D shaders and media/gpgpu kernels. Internally each is capable of multi-issue SIMD execution, and their hardware multi-threaded operation provides a very high-efficiency execution environment in the face of long data latencies typically associated with memory accesses. Each hardware thread within an EU has a dedicated large-capacity high-bandwidth register file (GRF) and associated independent thread-state. Execution is multi-issue per clock to pipelines capable of integer, single and double precision floating point operations, SIMD branch capability, logical operations, transcendental operations, and other miscellaneous operations. Communication to support units (shared functions) for operations such as texture sampling or scatter/gather load/stores is via 'messages' programmatically constructed and 'sent' to those functions, with dependency hardware causing the issuing thread to sleep until the requested data has been returned.

EU instance count varies by product generation, as well as by SKU within a given generation, and their capabilities have evolved over the many generation of the Architecture. Please see "Device Attributes" in the "Configuration" chapter for specific rates and capacities associated with Execution Units.

Shared Functions

Shared functions are hardware units that provide specialized supplemental functionality for the EUs. A shared function is implemented where the demand for a given specialized function is insufficient to justify the costs on a per-EU basis. Instead a single instantiation of that specialized function is implemented as a stand-alone entity outside the EUs and shared among the EUs.

Invocation of the shared functionality is performed via a communication mechanism called a message. A message is a small self-contained packet of information created by a kernel and directed to a specific shared function.

Programming Note	
Context:	Communication mechanism in shared functions
The message is defined by a range of Message Register File (MRF) registers that hold message operands, a destination shared function ID, a function-specific encoding of the desired operation, and a destination General Register File (GRF) register to which any writeback response is directed.	

Messages are dispatched to the shared function under software control via the *send* instruction. This instruction identifies the contents of the message and the GRF register locations to direct any response.

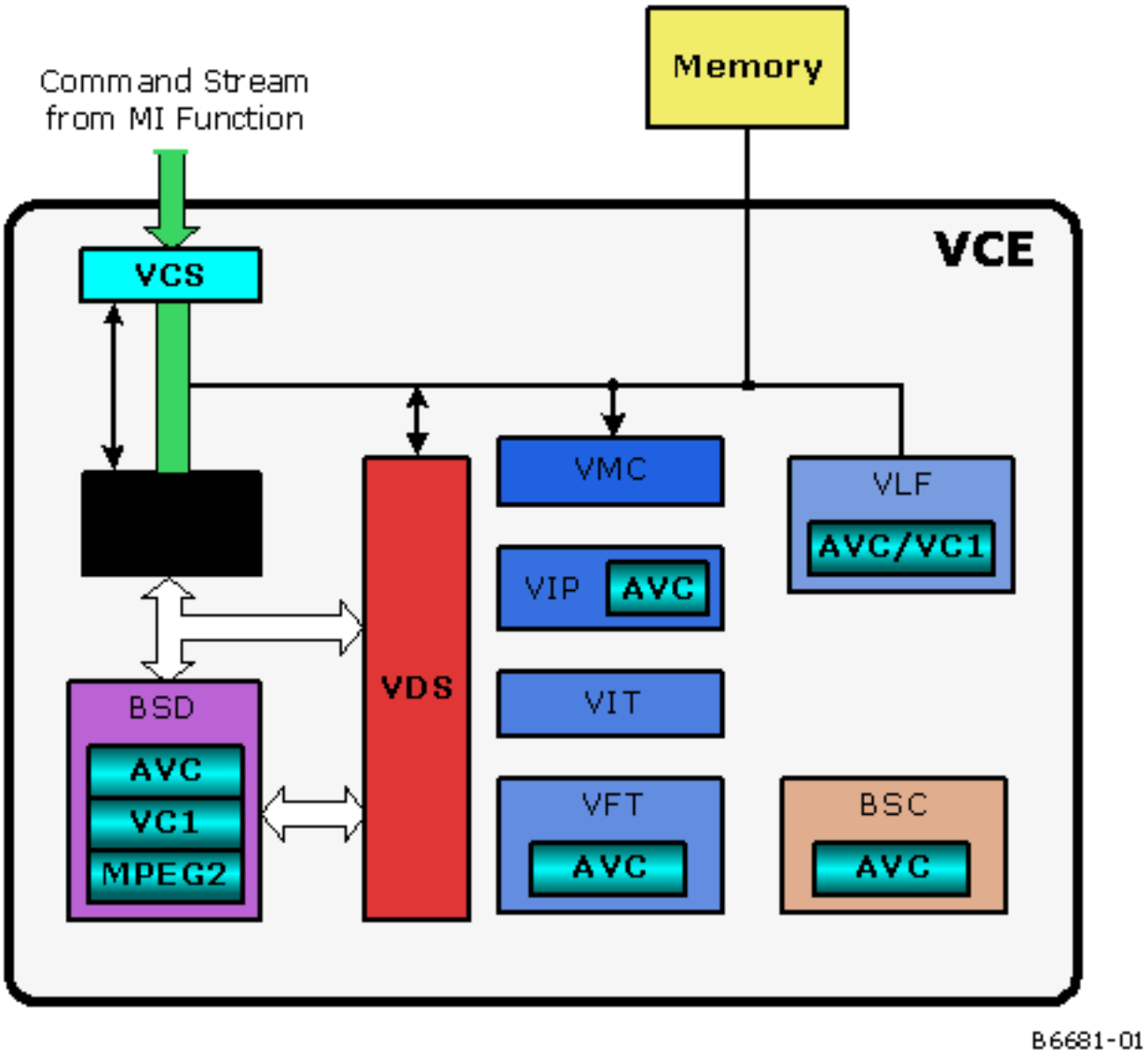
The message construction and delivery mechanisms are general in their definition and capable of supporting a wide variety of shared functions.

Video Codec Engine

The parallel Video Codec Engine (VCE) is a fixed function video decoder and encoder engine. It is also referred to as the multi-format codec (MFX) engine, as a unified fixed function pipeline is implemented to support multiple video coding standards such as MPEG2, VC1, and AVC:

- VCS – VCE Command Streamer unit
- BSD – Bitstream Decoder unit
- VDS – Video Dispatcher unit
- VMC – Video Motion Compensation unit
- VIP – Video Intra Prediction unit
- VIT – Video Inverse Transform unit
- VLF – Video Loop Filter unit
- VFT – Video Forward Transform unit (encoder only)
- BSC – Bitstream Encoder unit (encoder only)

VCE Diagram



Device	AVC BSD	VC1 BSD	AVC Dec	VC1 Dec	MPEG2 Dec	AVC Enc
	No	No	Yes	Yes	Yes	Yes