Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®, Pentium® Gold Processors based on the "Tiger Lake" Platform

Volume 6: Memory Views

May 2023, Revision 2.0

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, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of 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.

Revision History

Revision	Description
1.0	Initial Release
2.0	The following sections were added: • Memory Types and Cache Interface • Memory Object Control State (MOCS) • L3 Control Registers • Memory Interface Control Registers • Required PAT & MOCS Tables • LNCFCMOCSx • GLOB_MOCS_LECC_x

Table of Contents

Memory Views

Introduction

A modern GPU consists of multiple "engines", including Compute, Render (including Fixed Functions), Media Encode/Decode, Media Enhancement, Blitter/Copy, etc. Engines can operate concurrently and independently using different virtual address spaces.

All engines rely heavily on access to and from memory resources to perform their various functions. The memory subsystem connects engines to the memory resources, and provides services such as address translation, compression, caching, and HW virtualization. The memory subsystem is the heart of the GPU.

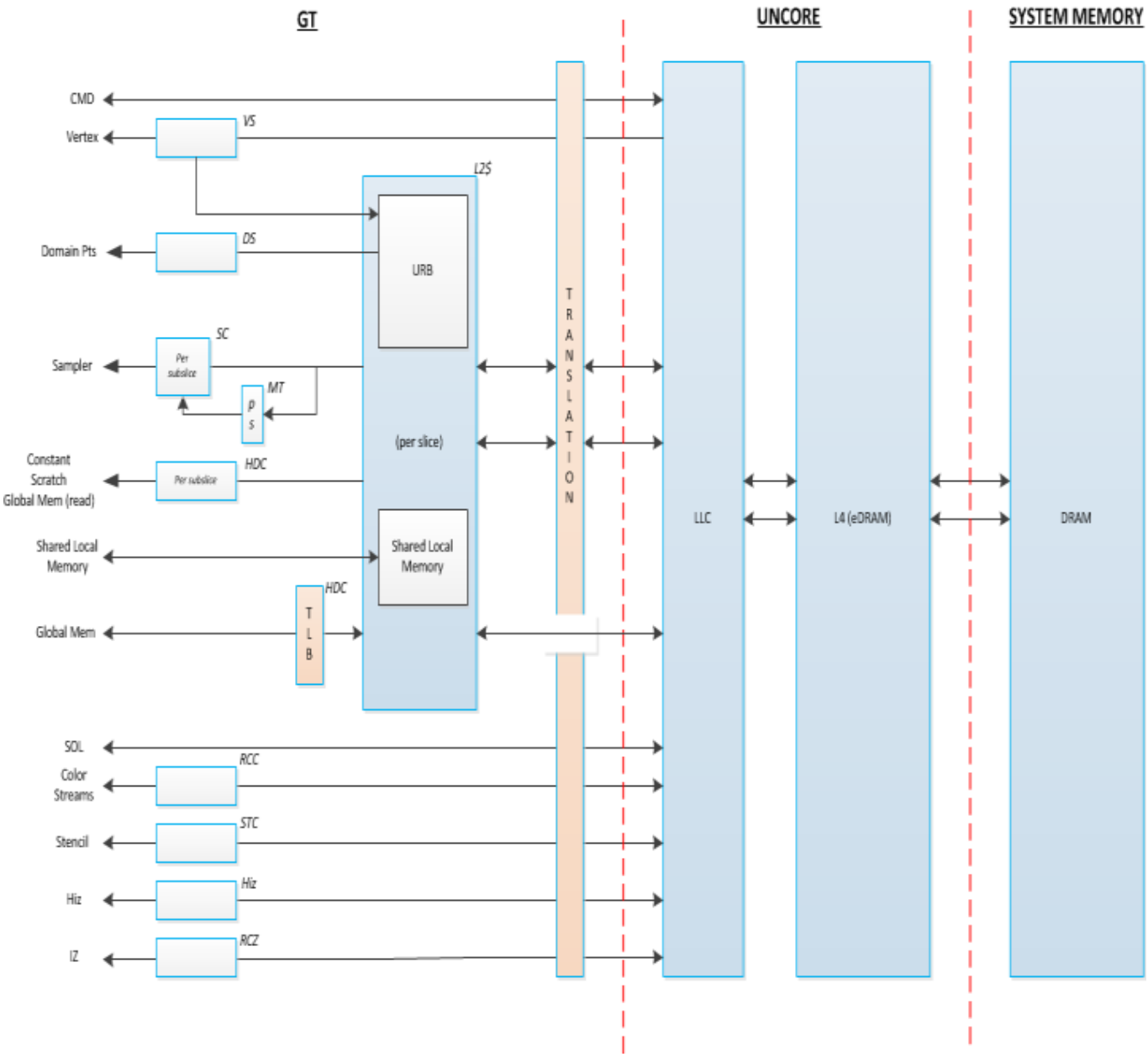
Key components of the memory subsystem include:

- Translation Services : Translation Lookaside Buffer (TLB) that translates virtual addresses associated with a context (process) running on an engine, to the physical address space of the GPU or System, and performs page table walks if a request misses in the TLB
- L3 Cache : Level 3 cache that is shared by all engines within the "GT" domain. Engines may have their own Level 1 and Level 2 caches that are not shared.
- LLC : Last Level Cache of the CPU host (only relevant for Integrated GPU)
- L4 Cache : Optional cache between L3 and Memory
- Compression : Handles lossless compression and decompression of memory objects
- System Memory : Memory that is physically attached to the CPU and managed entirely by the Operating System or Hypervisor
- Device Memory : Memory that is physically attached to a discrete GPU, or "stolen" from System Memory for an Integrated GPU, and managed entirely by the GPU device driver

The memory subsystem supports flows that are coherent with CPU memory (including CPU caches), as well as those that are not coherent with CPU memory. In general, non-coherent flows provide higher bandwidth more efficiently than coherent flows, but may require special handling by SW.

The following diagram provides an overview of a typical memory subsystem an Integrated GPU.

Cache and Memory Hierarchy



Graphics Virtual Memory

The GPU uses a virtual memory address space, where the graphics virtual address is mapped through a Page Table to a physical memory address. Normally, this mapping is set up by the graphics device driver and is private to the GPU context. However, in some cases the graphics virtual address is shared with the CPU - see for more information.

The range of valid graphics virtual addresses, and the types of page tables supported for address translation, varies with the GPU configuration. See the section for a summary the ranges and features supported by a specific graphics device.

Although the range of supported graphics virtual addresses varies, most GPU commands and GPU instructions use a common 64 bit definition for a graphics virtual address. Addresses outside of the supported range are reserved for future address space expansion. See the **GraphicsAddress** structure definition for specific details.

Some GPU devices support an extended graphics virtual memory address mapping called Tiled Resources. When enabled, the Tiled Resources Translation Table (TR-TT) pre-processes graphics virtual addresses. TR-TT maps a graphics virtual memory address either to a new graphics virtual memory address or to a Null Tile. Null Tiles return zero on reads and drop writes. For translations that are not Null Tiles, the new graphics virtual memory address is then used for the graphics virtual address and translated through the normal Page Table to generate a physical memory address.

Graphics Translation Tables

The GPU supports standard virtual memory models as defined by the IA programmer's guide. This section describes the different paging models, their behaviors, and the page table formats.

The Graphics Translation Tables (GTT) are memory-resident page tables containing an array of Page Translation Entries (PTEs) used in mapping graphics virtual addresses to physical memory addresses. There are two types of page tables: Global GTT and Per-Process GTT.

The base address of the GGTT and the PPGTT are programmed via the PGTBL_CTL and PGTBL_CTL2 MI registers, respectively. The translation table base addresses must be 4KB aligned. The GGTT size is 8MB, to cover 4GB of Global Virtual Address space, and is physically contiguous (ie, "flat"). The global GTT should only be programmed via the MMIO range within the GTTMMADR BAR. The PPGTT is programmed directly in memory and is multi-level. The page tables are further described in later sections.

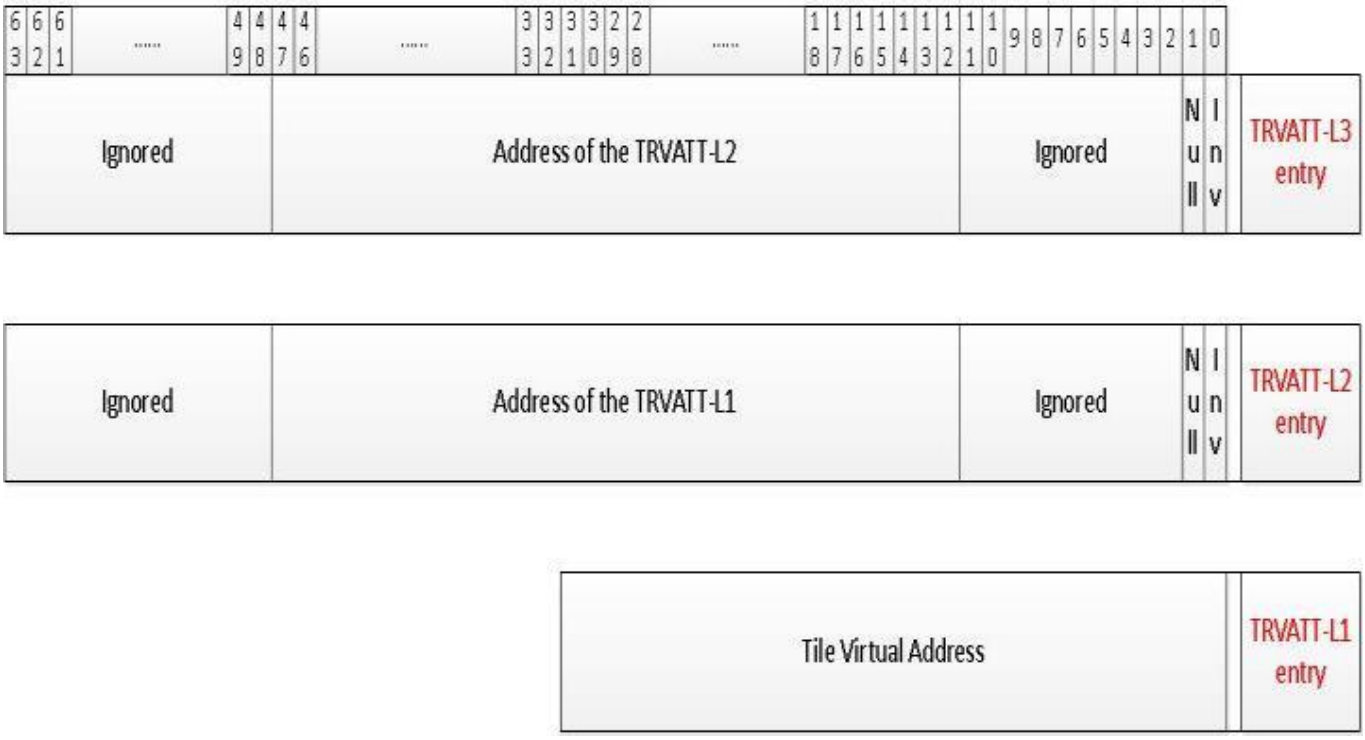
GFX Page Tables

This section describes the different types of address translation tables used by the GPU.

Tiled Resources Translation Tables

Sparse Tiled Resources can be thought of as a kind of application-controlled virtual memory scheme. The application allocates a resource in a virtual address space. Then the application tells the driver to map specified 64KB tiles within the surface to memory, within resources called Tile Pools. Tiles that are not mapped to a Tile Pool are null tiles.

Tiled Resource Translation Table (TRTT) is constructed as a 3 level tile Table. Each tile is 64KB in size which leaves behind 44-16=28 address bits. 28bits are partitioned as 9+9+10 which corresponds to TRVATT L3, L2 and L1 respectively. This is where TRVATT L3 has 512 entries, L2 has 512 entries and L1 has 1024 entries where each level is contained within a 4KB page hence L3 and L2 is composed of 64b entries and L1 is composed of 32b entries.



The contents of the TRVATT tables are as listed above where L3 and L2 points to the address of the next level which is a 4KB page and L1 contains the 32b VA address pointer needed to map the TR tile to virtual address space.

L1 Entry:

Bits	Field	Description
31:0	ADDR: Address	GFX virtual address of 64KB tile is referenced by this entry. This field is treated as GFX Virtual Address (GVA) when translated and maps to 47:16.

L2 Entry:

Bits	Field	Description
63:48	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
47:12	ADDR: Address	GFX virtual address or Guest Physical Address of 4KB base address pointing to TR-TT L1. *TR-TT table entries for L2 and L3 can be in GFX virtual address mode or Guest Physical address mode chosen by GFX software.*
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	Null	Null Tile where reads to this tile returns zero with a Null indicator and writes are dropped.
0	Invalid	Invalid Tile where reads to this tile returns zero and writes are dropped. Additional interrupt is generated to GFX software when an invalid tile is accessed.

L3 Entry:

Bits	Field	Description
63:48	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
47:12	ADDR: Address	GFX virtual address or Guest Physical Address of 4KB base address pointing to TR-TT L2. *TR-TT table entries for L2 and L3 can be in GFX virtual address mode or Guest Physical address mode chosen by GFX software.*
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	Null	Null Tile where reads to this tile returns zero with a Null indicator and writes are dropped.
0	Invalid	Invalid Tile where reads to this tile returns zero and writes are dropped. Additional interrupt is generated to GFX software when an invalid tile is accessed.

Programming Note	
Context:	Tiled ResourceTranslation Tables in Gfx Page Tables
GFX Driver has to disable the TR-TT bypass mode before using tiled resources translation tables. Details of the registers are given in "registers for TR-TT management."	

Programming Note	
Context:	Tiled ResourceTranslation Tables in Gfx Page Tables
GFX Driver is not allowed to put TR-TT entries into TR-VA space.	

Programming Note	
Context:	Tiled ResourceTranslation Tables in Gfx Page Tables
Usage model for TR translations are restricted to GFX Render Engine (& POSH pipeline).	

Programming Note	
Context:	Tiled ResourceTranslation Tables in Gfx Page Tables
TRTT is only for PPGTT64 (Advanced or Legacy PPGTT64). Enabling TRTT in Legacy PPGTT32 context or GGTT context is considered as invalid programming.	

Programming Note	
Context:	Tiled ResourceTranslation Tables in Gfx Page Tables
When partitioned address space based Dual Context is enabled, bit[47] of the virtual address in L3, L2 and L1 entries must be 0.	

Registers for TR-TT Management

Following register is a global mechanism to disable the bypass mode which is considered to be default for h/w. GFX driver has to set this bit to disable bypass mode before using TR-TTs.

Following registers shall be part of the h/w context.

Tiled Resources VA Translation Table L3 Pointer		
Register Space:		MMIO: 0/2/0
DWord	**Bit**	**Description**
1	63:48	**Reserved** **Access:** RO Reserved.
	47:32	**Tiled Resource - VA translation Table L3 Pointer (Upper Address)** **Default Value:** 0000h **Access:** R/W Upper address bits for tiled resource VA to virtual address translation L3 table. For physical memory option, address bits [47:39] has to be programmed to "0" as it is defined the limit of physical memory allocation.
0	31:16	**Tiled Resource - VA translation Table L3 Pointer (Lower Address)** **Default Value:** 0000h **Access:** R/W Lower address bits for tiled resource VA to virtual address translation L3 table.
	15:0	**Reserved** **Access:** RO Reserved.

Tiled Resources Null Tile Detection Register		
Register Space:		MMIO: 0/2/0
DWord	**Bit**	**Description**
	31:0	**Null Tile Detection Value** **Default Value:** 00000000h **Access:** R/W A 32bit value programmed to enable h/w to perform a match of TR-VA TT entries to detect Null Tiles. Hardware will flag each entry and space behind it as Null Tile for matched entries.

Tiled Resources Invalid Tile Detection Register		
Register Space:		MMIO: 0/2/0
DWord	**Bit**	**Description**
	31:0	**Invalid Tile Detection Value** **Default Value:** 00000000h **Access:** R/W A 32bit value programmed to enable h/w to perform a match of TR-VA TT entries to detect Invalid Tiles. Hardware will flag each entry and space behind it as Invalid Tile for matched entries.

Tiled Resources Virtual Address Detection Registers (TRVADR)		
Register Space:		MMIO: 0/2/0
DWord	**Bit**	**Description**
0	31:8	**Reserved** **Access:** RO Reserved.
	7:4	**TRVA Mask Value (TRVAMV)** **Default Value:** 0000b **Access:** R/W 4bit MASK value that is mapped to incoming address bits[47:44]. MASK bits are used to identify which address bits need to be considered for compare. If particular mask bit is "1", mapping address bit needs to be compared to DATA value provided. If "0", corresponding address bit is masked which makes it don't care for compare (*this field defaults to "0000" to disable detection*)

Tiled Resources Virtual Address Detection Registers (TRVADR)		
		Note that h/w supports two possible values for MASK: "0000" which is disabled case and "1111" where 44 bit TR-VA space is carved out.
	3:0	**TRVA Data Value (TRVADV)** **Default Value:** 0b **Access:** R/W 4bit DATA value that is mapped to incoming address bits[47:44]. Data bits are used to compare address values that are not filtered by the TRVAMV for match.

Tiled Resources Translation Table Control Register (TRTTE)		
Register Space:		MMIO: 0/2/0
DWord	**Bit**	**Description**
0	31:2	**Reserved** **Access:** RO Reserved.
	1	**TR-VA Translation Table Memory Location** **Default Value:** 0b **Access:** R/W This fields specifies whether the translation tables for TR-VA to VA are in virtual address space vs physical (GPA) address space. 0: Tables are in Physical (GPA) Space 1: Tables are in Virtual Address Space **Tiled Resource Translation Tables in GPA space is not supported in any generations. This mode should never be set as GPA mode (always set to '1). HW will set TRTT tables in Virtual address space mode only.**
	0	**TR-TT Enable** **Default Value:** 0b **Access:** R/W TR translation tables are disabled as default. This field needs to be enabled via s/w to get TR translation active.

The following register (0x4DFC[0]) has enable and disable control of the bypass path across TR translations. By default, bypass is enabled, and bypass needs to be disabled (by setting 0x4DFC[0] = '1) for TR translations to function. Disabling the bypass should be done before render power gating is enabled.

Detection and Treatment of Null and Invalid Tiles

Two types of definition that need to be extracted from TR-VA walk in addition to reaching the GFX virtual address.

1. **Null Tiles**: Null tiles provide the applications the of capability to preventing OS mapping the entire surface. When a memory access hits a Null tile, the access is terminated and zero's are returned to the originator of the memory access for loads along with a null indicator and for stores the access is dropped at the page walker level.
2. **Invalid Tiles**: This is the case where GFX software did not update the value of the mapping properly for hardware to separate resident vs null tiles. The Invalid Tile treatment is exactly same however additionally a unique interrupt is generated in h/w

Both detections are done by GPU:

- For L2/L3 entries, Null and Invalid tile information is already embedded in the TR-TT entries
- For L1 entries, the contents (32bits) are compared in hardware to pre-programmed values by GFX software (*values are provided in GFX MMIO space*). For the match values, two separate 32b registers are defined, one for Null Tile detection and one for Invalid Tile detection.

Hardware walking matching the value or detecting L2/L3 would terminate the walk (i.e. rest of the tables are not valid) and define the access as either Null or Invalid.

Programming Note	
Context:	Detection and treatment of null and invalid tiles.
The software is not allowed to program both Null and Invalid values to be the same.	

Programming Note	
Context:	TileX Surfaces and Null Tiles
NULL or Invalid Tiles are not supported on TileX surfaces.	

GPU implements a counter mechanism to roll-up the Null tile accesses detected. The counter value is exposed to GFX software via GFX MMIO.

In implementation, when the TR translation tables are in virtual address domain, the pages faults encountered while walking the IA32e pages are not reported back to the TR walkers or TLBs. These faults are handled as fault & halt, making these faults transparent to the TR walkers. However, when such a fault is not fixed (unsuccessful fault response) or when a non-recoverable fault encountered, main page walker HW converts the cycle to an invalid cycle. Thus, in this case, TR walker or TR TLBs will get incorrect read return data without any notification of the non-recoverable fault condition. Thus, TR walker/TLBs will continue with the TR-walk with incorrect data. This can lead to spurious cycles being generated. However, a Gfx reset/FLR is expected as a result of the non-recoverable fault.

TR-TT Modes

The L3 table pointer along with TRTTL3e/TRTTL2e is projected to support two modes of address space. Original intent was to have the contents to be in Virtual Address space (OS managed) and have them to be translated to GPA to HPA before getting accessed. Such mechanism will incur high latency penalties due to nested page translations. GPU shall have an additional mode where tiled-resources translation tables are in physical address space (GPA) and eliminate the need to have nested translations to reduce the potentially high miss latencies.

TR-TT walker shall have both modes supported. The Mode bit will be part of the same Register that provides TR-VA TT L3 pointer.

Memory Types and Cache Interface

This section has additional information on the types of memory which are accessible via the various GT mechanisms. It includes discussion on how the various paging models are used and accessed. See the Graphics Translation Tables for more detailed discussions on paging models.

This section also includes descriptions of how different surface types (MOCS) can be cached in the L3 and the different behaviors which can be enabled.

Memory Object Control State (MOCS)

The memory object control state defines the behavior of memory accesses beyond the graphics core, including encryption, graphics data types that allow selective flushing of data from outer caches, and controlling cacheability in the outer caches.

This control uses several mechanisms. Control state for all memory accesses can be defined page by page in the GTT entries. Memory objects that are defined by state per surface generally have additional memory object control state in the state structure that defines the other surface attributes. Memory objects without state defining them have memory object state control defined per class in the STATE_BASE_ADDRESS command, with class divisions the same as the base addresses. Finally, some memory objects only have the GTT entry mechanism for defining this control. The table below enumerates the memory objects and the location of the control state for each:

Memory Object	Location of Control State
surfaces defined by SURFACE_STATE: sampling engine surfaces, render targets, media surfaces, pull constant buffers, streamed vertex buffers	SURFACE_STATE
depth, stencil, and hierarchical depth buffers	corresponding state command that defined the buffer attributes
stateless buffers accessed by data port	STATE_BASE_ADDRESS
indirect state objects	STATE_BASE_ADDRESS
kernel instructions	STATE_BASE_ADDRESS
push constant buffers	3DSTATE_CONSTANT_(VS | GS | PS)
index buffers	3DSTATE_INDEX_BUFFER
vertex buffers	3DSTATE_VERTEX_BUFFERS
indirect media object	STATE_BASE_ADDRESS
generic state prefetch	GTT control only
ring/batch buffers	GTT control only
context save buffers	GTT control only
store DWord	GTT control only

MOCS Registers

These registers provide the detailed format of the MOCS table entries, that need to be programmed to define each surface state.

MEMORY_OBJECT_CONTROL_STATE

Size (in bits): 7

Default Value: 0x00000000

DWord	Bit	Description
0	6:1	**Index to MOCS Tables** The index to define the L3 and system cache memory properties. The details of the controls are further defined in L3 and Page walker (memory interface) control registers. The field is defined to populate 64 different surface controls to be used concurrently. Related control registers can be updated during runtime. **Programming Notes** When an access is made through Data Port and the index to MOCS[6:1] = [48,59] (decimal), that surface or stateless memory access can be cached in HDC L1 cache. Accesses made through Data Port with MOCS[6:1] < 48 or > 59 will bypass the HDC L1 cache. This bypass is useful when software wants to ensure that Data Port access are coherent with the L3 memory.
	0	**Reserved**

L3 Control Registers

64x16b control registers are defined within L3 space to interpret MOCS indexing and map it to cache events.

The incoming MOCS value is used to index into one of these registers which hardware uses as control parameters for a given surface. It allows 64 concurrent surface definitions with unique control values for L3 caching.

Also attached are the default settings for each 64 locations if driver chooses to use as is.

Following 16b defines per selection definition:

Register#64 (MOCS value 63) is reserved for h/w use and should not be used by s/w.

In L3 Node: B020-B09F (128 Bytes). Please refer to the register section for default values.

Bits	Description
16:6	Reserved.
5:4	**L3 Cacheability Control (L3CC).** Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls. 00b: Use binding table index for direct EU accesses - for rest it is reserved. 01b: Uncacheable (UC) - non-cacheable. 10b: Reserved 11b: Writeback (WB).
3:1	**Skip Caching Control (SCC).** Defines the bit values to enable caching. Outcome overrides the L3caching for the surface. If "0" - than corresponding address bit value is don't care. Bit[1]=1: Address bit[9] needs to be "0" to cache in target. Bit[2]=1: Address bit[10] needs to be "0" to cache in target. Bit[3]=1: Address bit[11] needs to be "0" to cache in target.
0	**Enable Skip Caching (ESC).** Enable for the Skip cache mechanism. 0: Not enabled. 1: Enabled for L3.

Memory Interface Control Registers

64x32b control registers are defined within the page walker where control parameters for LLC/eDRAM caching are defined. Incoming memory control object state index is used to do a look up into the table where the corresponding control parameters are picked for a given surface. These control values are used to control LLC/eDRAM caching.

For EU surfaces where binding table index is used, we also pass two bits of information in the hardware.

Following 32b defines per selection definition:

All MOCS registers are global and need to be saved/restored on RC6 entry/exit.

Certain MOCS indices are used by hardware and carry special meanings. These indices are restricted from being used for regular surfaces. In the following list, the indices are 0 based (i.e., MOCS index 0 is the first register and MOCS index 63 is the last register.)

- Index 'd63 is used for two purposes.
  - It is used by the L3 for all its evictions. The programming of the index 'd63 is expected to allow LLC cacheability to enable coherent flows to be maintained.
  - It is also used by hardware to force L3 uncacheable cycles. The programming of the index #63 is expected to make the surface L3 uncacheable.
- Index 'd62
  - This index is used for Tiled-Resources page walker accesses in previous projects.
  - This index is used for AuxTT Nodes for current projects.
- Index 'd61 is used for displayable surfaces. The programming of the index 'd61 is expected to disallow LLC cacheability for the surface to be displayable.
- Index 'd60 is reserved for use in the 3D CCS accesses.

Bits	Description
31:19	**Reserved**
18:17	**Self Snoop Enable** 00: Default value. Self snoop attribute sent to the uncore is as normal - determined by MIDI unit logic 01: Override the self snoop bit generated by MIDI with 0. No self snoops are sent to the uncore for any transactions from this surface 11: Override the self snoop bit generated by MIDI with 1. Self snoops are always sent to the uncore for any transactions from this surface
16:15	**Class of Service** This field controls the Class of Service sent to the LLC to determine which sub-set of Ways the surface will be stored in. The allocation of certain LLC ways to different class of service settings is a project dependent decision and listed in the Bspec. 00: Value from Private PAT Registers(40E0/40E4/40E8/40EC) 01: Class 1 10: Class 2

Bits	Description
	11: Class 3
14	**Snoop Control Field (SCF)**: Enables s/w to have GFX h/w to be able to consume IA generated buffers that are tagged as WB. Driver can mark these buffers as WB when generating them from IA. In LP-SOCs, the fabric is not forced to be coherent all the time. IA-core generated WB buffers can only be consumed by GPU if that buffer is tagged as snoop-able in GPUs buffer definitions (or via GPU Page tables). 1: Non-Coherent Write/Read 0: Coherent Access
13:11	**Page Faulting Mode** This fields controls the page faulting mode that will be used in the memory interface block for the given request coming from this surface: 000: Use the global page faulting mode from context descriptor (default) 001-111: Reserved
10:8	**Skip Caching Control** Defines the bit values to enable caching. Outcome overrides the LLC caching for the surface. If "0" - than corresponding address bit value is do not care Bit[8]=1: address bit[9] needs to be "0" to cache in target Bit[9]=1: address bit[10] needs to be "0" to cache in target Bit[10]=1: address bit[11] needs to be "0" to cache in target The default value of this field is '000. **For coherent surfaces, skip caching should not be enabled, as not caching in LLC breaks the coherency.**
7	**Enable Reverse Skip Caching** Enable for the Skip cache mechanism 0: Not enabled 1: Enabled for LLC
6	**Don't Allocate on miss** Controls defined for RO surfaces in mind, where if the target cache is missed - do not bring the line (applicable to LLC/eDRAM). 0: Allocate on MISS (normal cache behavior) 1: Do NOT allocate on MISS
5:4	**LRU (Cache Replacement) Management (LRUM).** This field allows the selection of AGE parameter for a given surface in LLC or eLLC. If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1",

Bits	Description
	or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. 00: Take the age value from Uncore CRs. 01: Assign the age of "0" 10: Don't change the age on a hit. 11: Assign the age of "3"
3:2	**Target Cache (TC).** This field allows the choice of LLC vs eLLC for caching. 00b: Use TC/LRU controls from page table 01b: LLC Only. 10b: LLC/eLLC Allowed. 11b: LLC/eLLC Allowed. **For coherent surfaces ensure that LLC caching is enabled - even when using target cache controls from page table.**
1:0	**LLC/eDRAM Cacheability Control (LeCC).** Memory type information used in LLC/eDRAM. 00b: Use Cacheability Controls from page table / UC with Fence (if coherent cycle). 01b: Uncacheable (UC) - non-cacheable. 10b: Writethrough (WT). 11b: Writeback (WB). **Note:** In case of SVM (advanced context), LLC/eDRAM memory type is used based on the page table controls and cannot be managed via MOCS index.

Defaults Table

Default	LeCC	TC	LRUM	AOM	ESC	SCC	PFM	Default	LeCC	TC	LRUM	AOM	ESC	SCC	PFM
000000	00	00	11	0	0	00	000	000000	00	00	11	0	0	00	000
000001	00	01	11	0	0	00	000	000001	00	01	11	0	0	00	000
000010	00	10	11	0	0	00	000	000010	00	10	11	0	0	00	000
000011	01	00	11	0	0	00	000	000011	01	00	11	0	0	00	000
000100	10	00	11	0	0	00	000	000100	10	00	11	0	0	00	000
000101	10	01	11	0	0	00	000	000101	10	01	11	0	0	00	000
000110	10	10	11	0	0	00	000	000110	10	10	11	0	0	00	000
000111	11	00	11	0	0	00	000	000111	11	00	11	0	0	00	000
001000	11	01	11	0	0	00	000	001000	11	01	11	0	0	00	000
001001	11	10	11	0	0	00	000	001001	11	10	11	0	0	00	000
001010	10	00	11	0	0	00	000	001010	10	00	11	0	0	00	000
001011	10	01	11	0	0	00	000	001011	10	01	11	0	0	00	000
001100	10	10	11	0	0	00	000	001100	10	10	11	0	0	00	000
001101	11	00	11	0	0	00	000	001101	11	00	11	0	0	00	000
001110	11	01	11	0	0	00	000	001110	11	01	11	0	0	00	000
001111	11	10	11	0	0	00	000	001111	11	10	11	0	0	00	000
010000	00	00	11	0	0	00	000	010000	00	00	11	0	0	00	000
010001	00	01	11	0	0	00	000	010001	00	01	11	0	0	00	000
010010	00	10	11	0	0	00	000	010010	00	10	11	0	0	00	000
010011	01	00	11	0	0	00	000	010011	01	00	11	0	0	00	000
010100	10	00	11	0	0	00	000	010100	10	00	11	0	0	00	000
010101	10	01	11	0	0	00	000	010101	10	01	11	0	0	00	000
010110	10	10	11	0	0	00	000	010110	10	10	11	0	0	00	000
010111	11	00	11	0	0	00	000	010111	11	00	11	0	0	00	000
011000	11	01	11	0	0	00	000	011000	11	01	11	0	0	00	000
011001	11	10	11	0	0	00	000	011001	11	10	11	0	0	00	000
011010	10	00	11	0	0	00	000	011010	10	00	11	0	0	00	000
011011	10	01	11	0	0	00	000	011011	10	01	11	0	0	00	000
011100	10	10	11	0	0	00	000	011100	10	10	11	0	0	00	000
011101	11	00	11	0	0	00	000	011101	11	00	11	0	0	00	000
011110	11	01	11	0	0	00	000	011110	11	01	11	0	0	00	000
011111	11	10	11	0	0	00	000	011111	11	10	11	0	0	00	000

Required PAT & MOCS Tables

Rather than hard-wire the available PAT and MOCS Table entries, the tables are software programmable. But to reduce virtualization overhead, Intel requires that all drivers use the table values specified in this section. This minimizes virtualization overhead while still allowing the flexibility of post-silicon/post-launch improvements from new performance findings and updated tables.

To ensure software backward and forward compatibility, table entries are version-numbered, where

- An entry version indicates the version of the table in which it was introduced.
- Each platform starts with version 1 entries.
- Entries are only ever added (never modified or removed—except to correct documentation/etc. errors, if the change is backward-compatible).
- Added entries use previously highest version number, incremented by one.
- Multiple entries added at same time share same version number.
- Spec'ed entries never change versions—Version increases are only for addition of new slots.

Should table versions > 1 be necessary on a given platform, graphics drivers will be able to query to determine which table version is available on their current (potentially virtualized) platform. When new drivers find themselves on a platform with older tables, they should remap their new/unavailable entries back to the most appropriate matches in the available tables.

PAT:

Version	Group	Use	PAT Index	PAT_INDEX MEM_TYPE
1	LLC	WB	0	3
1		WC	1	1
1		WT	2	2
1	UC	UC	3	0

MOCS:

Version	Group	Use	MOCS Index	LNCFCMOCSx			GLOB_MOCS_LECC_x									
				ESC	SCC	L3CC	LeCC	TC	LRUM	DAoM	ERSC	SCC	PFM	SCF	CoS	SSE
1	Base	Error (Reserved for Non-Use)	0	0	0	3	3	1	3	0	0	0	0	0	0	0
1		Reserved	1													
1		L3 + LLC	2	0	0	3	3	1	3	0	0	0	0	0	0	0
1		Uncached	3	0	0	1	1	1	0	0	0	0	0	0	0	0
1		L3 (Read-Only*)	4	0	0	3	1	1	0	0	0	0	0	0	0	0
1		LLC	5	0	0	1	3	1	3	0	0	0	0	0	0	0
1	Age 0	LLC (Age 0)	6	0	0	1	3	1	1	0	0	0	0	0	0	0
1		L3 + LLC (Age 0)	7	0	0	3	3	1	1	0	0	0	0	0	0	0
1	Age: Don't Chg.	LLC (Age:DC)	8	0	0	1	3	1	2	0	0	0	0	0	0	0
1		L3 + LLC (Age:DC)	9	0	0	3	3	1	2	0	0	0	0	0	0	0
1	No AOM	LLC (No AOM)	10	0	0	1	3	1	3	1	0	0	0	0	0	0
1		L3 + LLC (No AOM)	11	0	0	3	3	1	3	1	0	0	0	0	0	0
1		LLC (No AOM; Age 0)	12	0	0	1	3	1	1	1	0	0	0	0	0	0
1		L3 + LLC (No AOM; Age 0)	13	0	0	3	3	1	1	1	0	0	0	0	0	0
1		LLC (No AOM; Age:DC)	14	0	0	1	3	1	2	1	0	0	0	0	0	0
1		L3 + LLC (No AOM; Age:DC)	15	0	0	3	3	1	2	1	0	0	0	0	0	0
1	Reserved	Reserved	16													
1		Reserved	17													
1	Self-Snoop	L3 + LLC (Self Snoop)	18	0	0	3	3	1	3	0	0	0	0	0	0	3
1	Skip Caching	L3 + LLC(12.5%)	19	0	0	3	3	1	3	0	0	7	0	0	0	0
1		L3 + LLC(25%)	20	0	0	3	3	1	3	0	0	3	0	0	0	0
1		L3 + LLC(50%)	21	0	0	3	3	1	3	0	0	1	0	0	0	0
1		L3 + LLC(75%)	22	0	0	3	3	1	3	0	1	3	0	0	0	0
1		L3 + LLC(87.5%)	23	0	0	3	3	1	3	0	1	7	0	0	0	0
1	Reserved	Reserved	24													
1		Reserved	25													
1	Special Indexes	HDC:L1 + L3 + LLC	**48**	0	0	3	3	1	3	0	0	0	0	0	0	0
1		HDC:L1 + L3	**49**	0	0	3	1	1	0	0	0	0	0	0	0	0
1		HDC:L1 + LLC	**50**	0	0	1	3	1	3	0	0	0	0	0	0	0
1		HDC:L1	**51**	0	0	1	1	1	0	0	0	0	0	0	0	0
1		Special Case (CCS)	**60**	0	0	1	3	1	3	0	0	0	0	0	0	0

1		Special Case (Displayable**)	**61**	0	0	3	1	1	0	0	0	0	0	0	0	0
1		HW Reserved—SW program but never use.	**62**	0	0	1	3	1	3	0	0	0	0	0	0	0
1		HW Reserved—SW program but never use.	**63**	0	0	1	3	1	3	0	0	0	0	0	0	0

HDC:L1 should only be used in read-only use-cases.

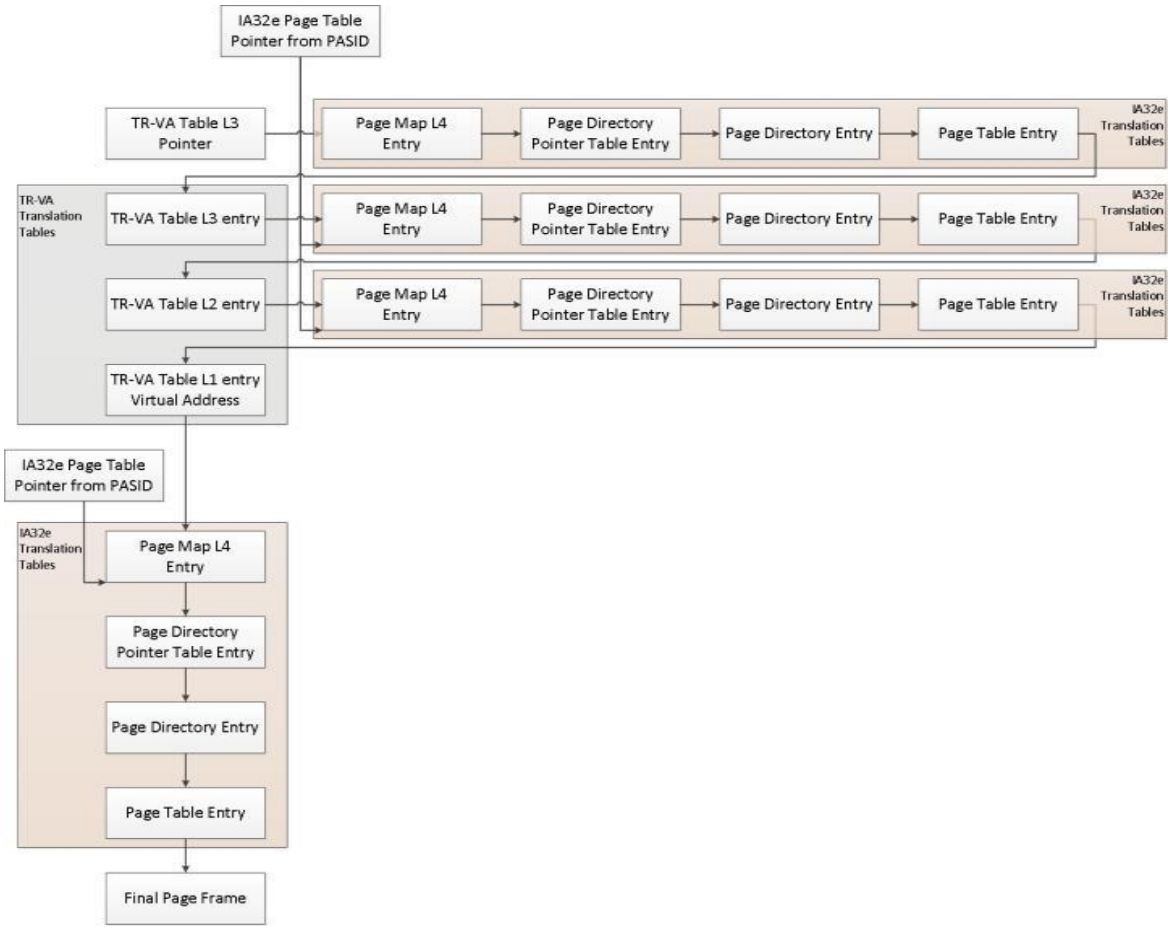
* Suitable solely for Read-Only use, since (written/dirty) L3 evictions essentially always go to LLC (via hard-wiring to MOCS#63), except for the MOCS#61 special-case.

** MOCS#61 is generally said to be for "Displayable" memory objects—though more precisely, it is for any memory object desiring writable L3 caching without LLC caching.

Virtual Addressed TR Translation Tables

Having sparse tiled resource translation tables in GFX virtual space requires the h/w TR-TT walker to walk thru the 1st level tile tables for table accesses to reach to Physical address at the L1 TR translation tables.

The following diagrams provide the view of the walk TR-VA translation tables are in physical memory and no 2nd Level (VTd) translations enabled.



Once 2nd level translations are enabled each level of 1st level walk needs to be further walked through VTd page tables.

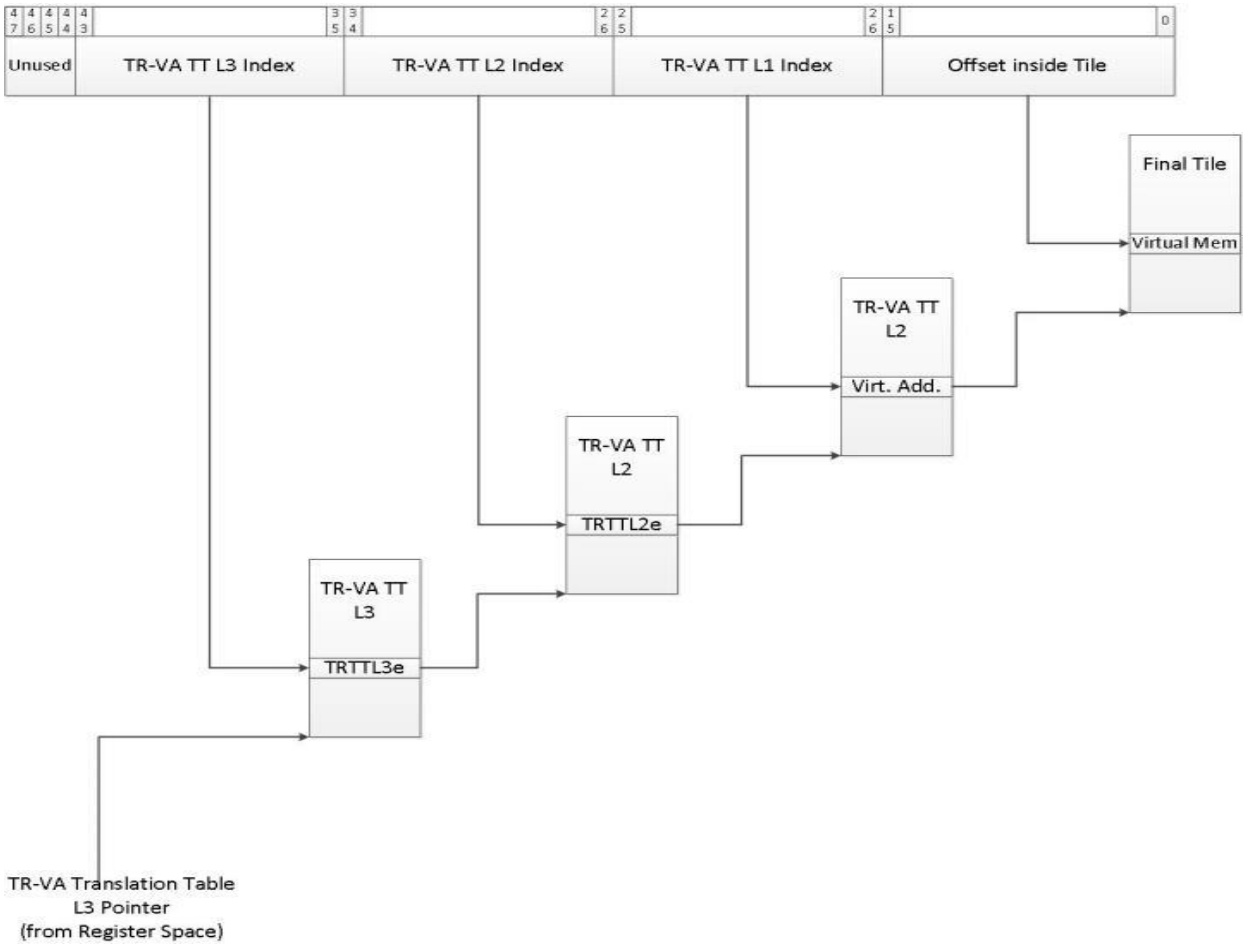
The level of nested walks does not change the structure of the TR-VA walker; it just defines the recursive nature of the translations.

TR-TT Page Walk

Sparse Tiled Resources translation tables are separated into 3-levels. The pointer to L3 table is going to be set up in GFX MMIO space as part of the context, this pointer be would be available to page walker ahead of any TR-VA memory accesses.

TR-TT L3 walk will be consistent of calculating the 64b of interest based on the L3 table pointer and using the 9 bit index (address bits[43:35]). L2 will use TR-TT L3 entry as the table pointer and use the next set of 9 address bits ([34:26]) to locate the L2 entry which is a pointer to L1 table. Final L1 table is located with L2 entry and indexed by remaining 10 address bits (25:16) to index where 32b virtual address is extracted.

Post TR-TT walk 32b entry from L1 is mapped to final virtual address 47:16 and remaining 15:0 is passed from the original TR-VA access as is given all tiles in TR-VA space are 64KB in size.



Page Table Modes

GFX Aperture and Display accesses are mapped thru Global GTT to keep the walk simple (i.e. 1-level) and latency sensitive. GPU accesses to memory can be mapped via Global GTT and/or ppGTT with various addressing modes.

Supported walk modes are listed as following:

1. **Global GTT with 32b virtual addressing**: Global GTT usage is similar to previous generations with extended capability of increasing virtual address (VA) up to 4GB (from 2GB) and use a standard 64b PTE format. The breakdown of the PTE for global GTT is given in later sections and allows 1-level page walk where the 20b index is used to select the 64b PTE from memory.
2. **Legacy 48b VA with ppGTT**: GFX address expansion beyond 4GB is added to address 48b virtual address space. 48b VA requires 36b indexing (4x9b) translating into 4-levels of page walk. To reduce the overhead of 4 level walk, GPU will cache the entire content of PML4 (4kB) to limit the on-demand walks to 3 levels. The caching happens as part of the initial demand where no further replacements required.

Global GTT

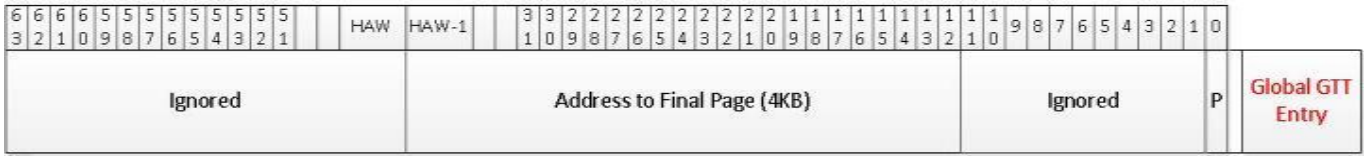
The GPU supports a Global Virtual Address space that is used for resources that are accessible to privileged (ie kernel-mode) processes, and not tied to a specific user-level process. For example, the Graphics micro-Controller (GuC) and Display Engine (if present) utilize Global address space. The Global VA space is 4GB in size (32-bit VA).

The Global GTT (GGTT) translates from the Global virtual address to a physical address that can be accessed by HW, using a page size of 4KB. The GGTT is a flat, single-level table. Each page table entry (PTE) in the GGTT is 8B in size, so the overall size of the GGTT 8MB.

The Global GGTT entries can be updated by the graphics driver, via the GTTMMADR MMIO range of the device, and also via the Command Streamer command MI_UPDATE_GTT.

Page Table Entry

The following page table entry will be used for Global GTT:



Bits	Field	Description
63:57	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
51:HAW	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	Address	Physical address of 4KB memory page referenced by this entry.
11:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4:2	Function	The Number of the Function to which this page has been assigned. This field is

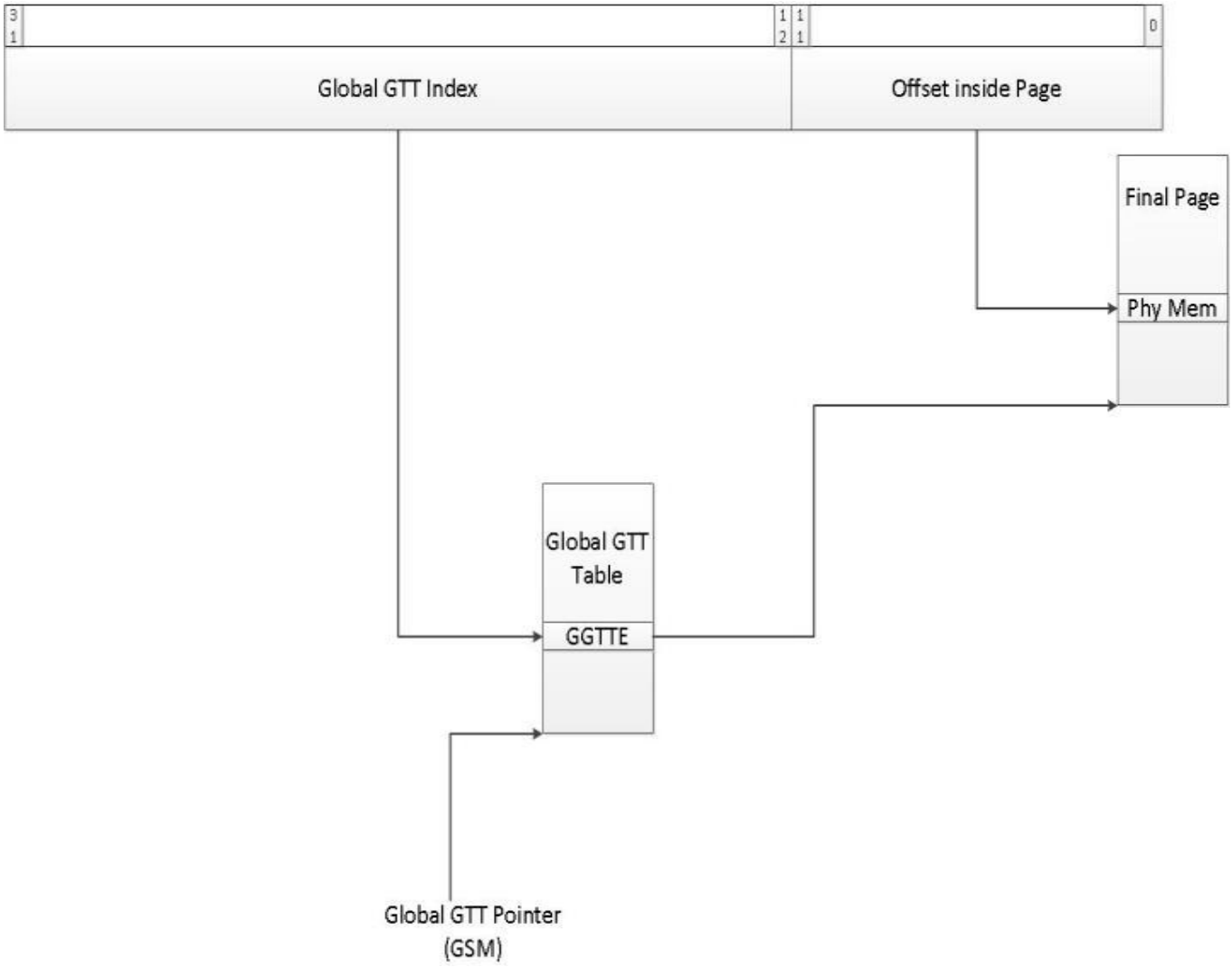
Bits	Field	Description
	Number	ignored.
1	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
0	Present	When set to 1, indicates that this Page Table Entry is Valid, and the corresponding page is Present in physical memory

* HAW = 39 for client, and 46 for server.

The GPU accesses GGTT table entries as uncacheable.

Page Walk

The global GTT page walk is identical to what it was prior. The only difference would be that each entry is 8B (instead of 4B) hence the entry selection needs to be updated once the corresponding Page Table miss read is returned.



Per-Process GTT with 48b VA

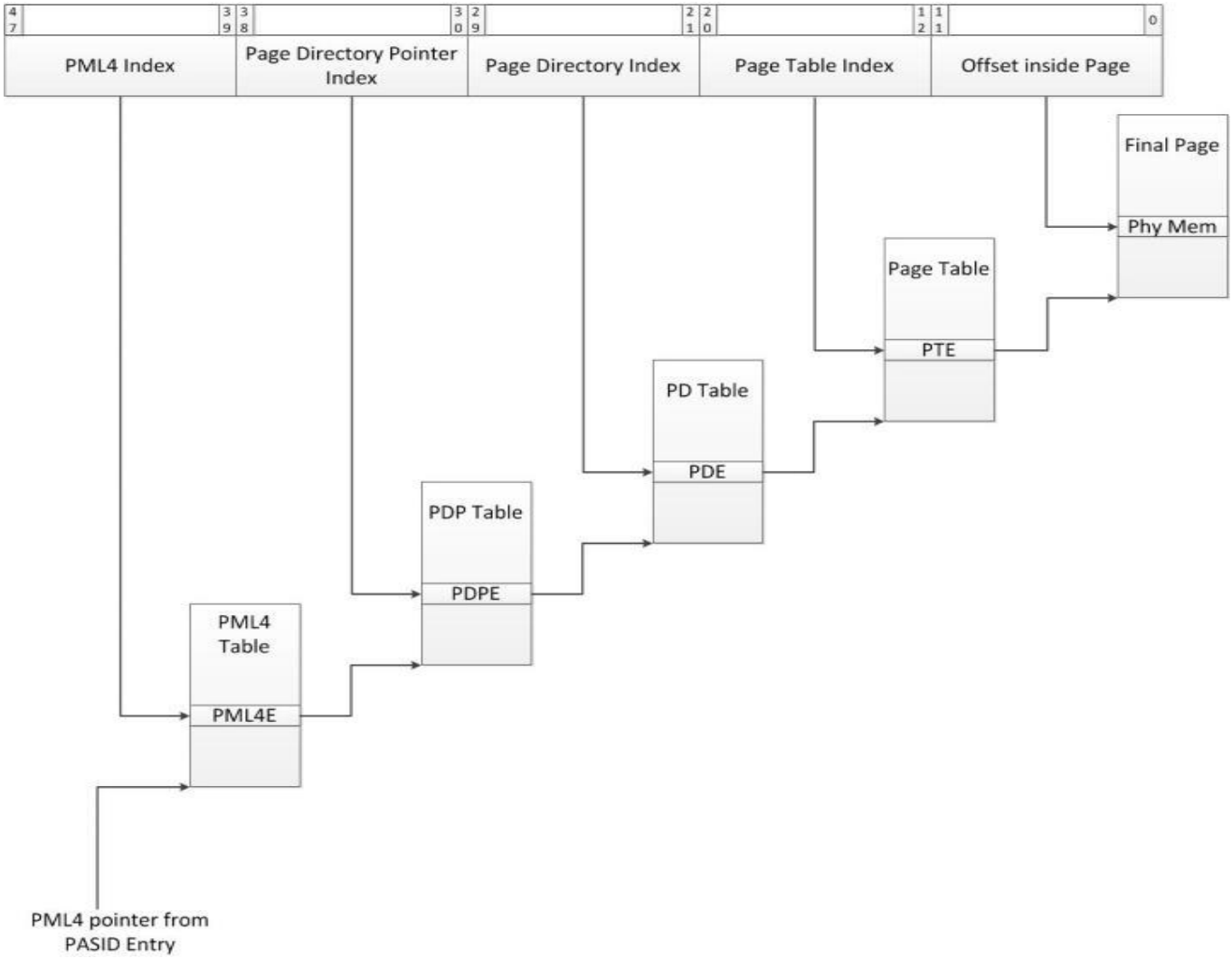
The GPU typically operates on behalf of user-level processes (applications), each of which has it's own "Per-Process" virtual address space. The size of this space is 256TB (48b address width).

The Per-Process GTT (PPGTT) translates these virtual addresses to physical addresses that are used by HW. The PPGTT is a multi-level table very similar to the IA32e table utilized by Intel CPUs. Each entry in the PPGTT is 8 Bytes.

Page Walk in Legacy 48b Mode

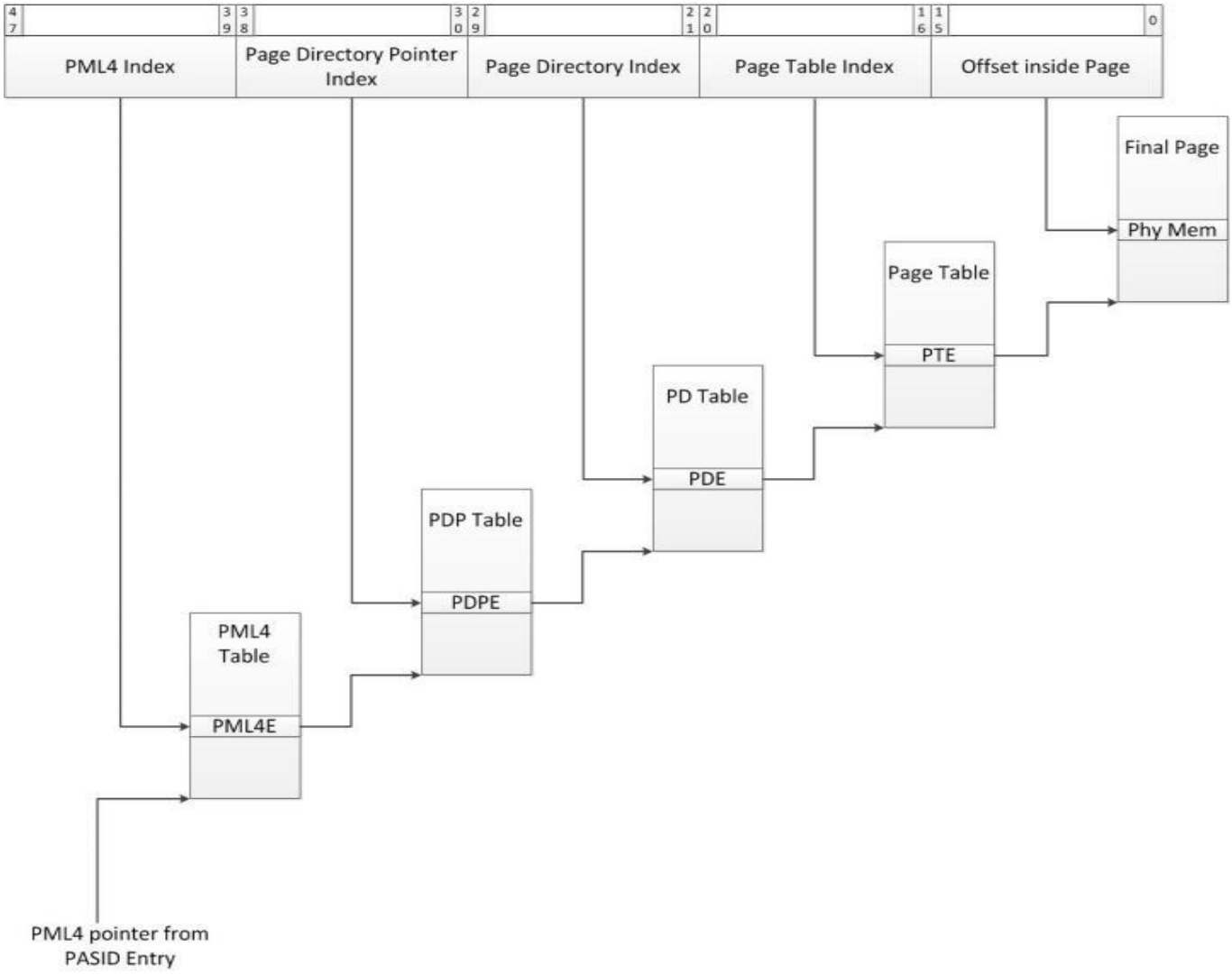
Translation of a 48b VA requires 4 levels of page table. assuming each table level is 4KB and each entry is 8B. The top-most level of the PPGTT is located via the PML4 table pointer associated with the process, and the 48b VA as used to index into consecutive levels of page table.

The following diagram shows the page walk that is needed for a 4KB page.



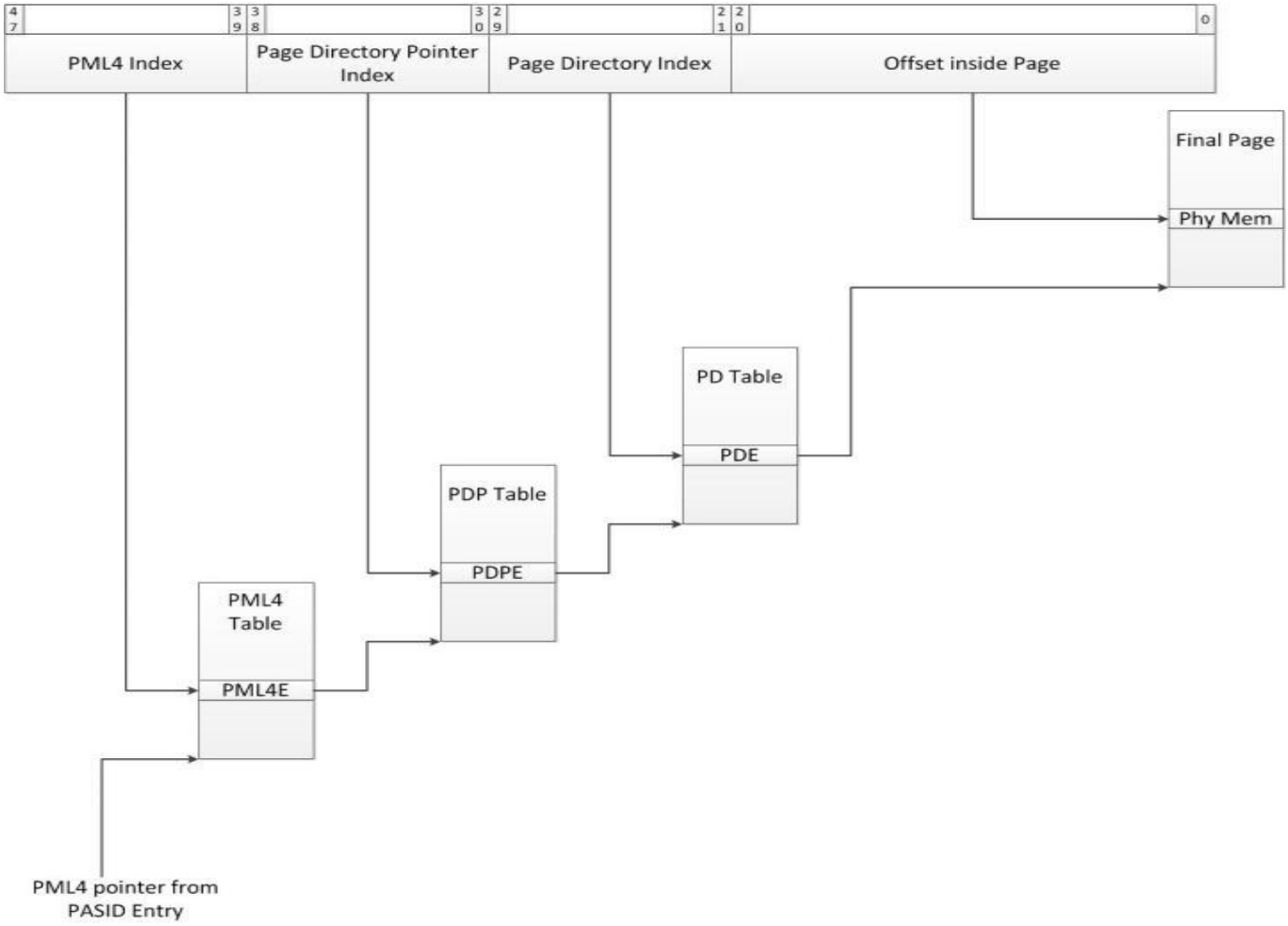
Walk with 64KB Page

64KB Page size has a slightly different usage for how PTEs are selected for the corresponding 64KB page. In page table every 16th entry (PTE#0, PTE#16, PTE#32....PTE#496) should be used to index. This is calculated using address [20:16]& "0000". Note that hardware should not make any assumptions for any other PTEs. 64K paging in the PTE is indicated by [11] of PDE. When PDE[11] = '1, every 16th PTE entry is read (by masking Adr[15:12] bits).



Walk with 2MB Page

With the 2MB Page walk, last level of the page walk is skipped where the PD entry points to the final page.



Walk with 1GB Page

1GB pages are supported by completing address translation at the PDP level, similar to how 2MB page translation is complete at PDE level.

Page Tables Entry PTE Formats

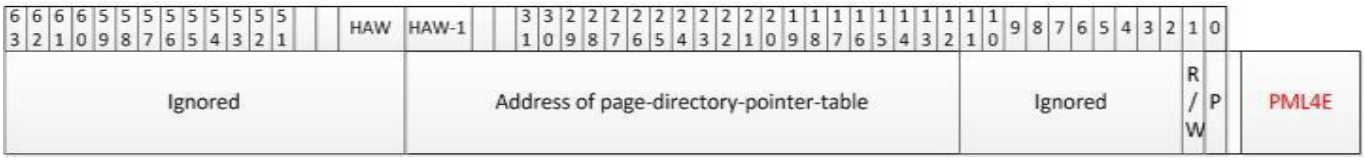
Page Table Entry (PTE) formats will follow the IA32e layout as given below:

Each table entry is further broken down along with the required functions. GFX has a 4 level page table which is pointed out by context descriptor starting with the PML4. The next levels have slightly different formats depending on the size of the page supported. 1GB and 2MB page formats are required for support.

Pointer to PML4 table

The pointer to PML4 table is provided via the context descriptor.

PML4E: Pointer to PDP Table



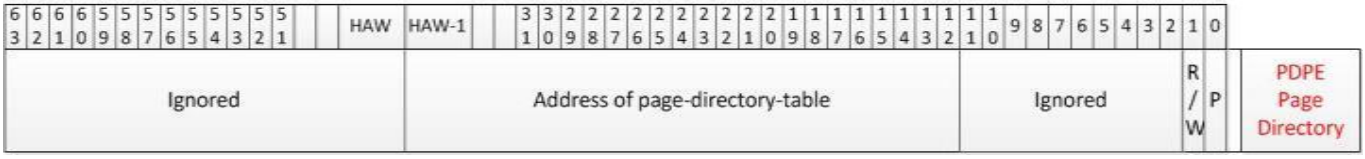
Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page-directory-pointer table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	PML4 Entry is present. It must be "1" to point to a page directory pointer table

* HAW = 39 for client, and 46 for server.

PDPE: Pointer to PD Table

PDP entry is used to locate the page directory. IA32e supports 1GB pages, the PDPE has a mechanism to identify a way to say whether this PDPE represents a pointer to page directory or to a contiguous 1GB physical memory.

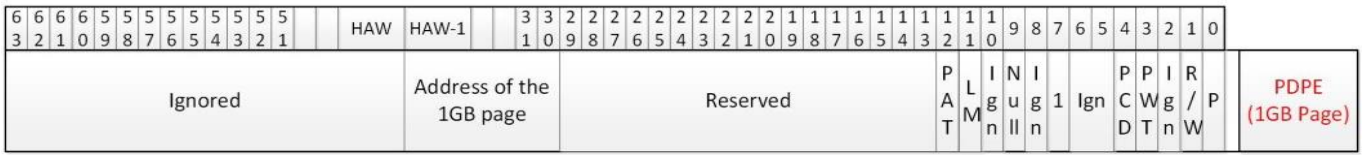
PDPE for PD



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page-directory table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. Access rights are described later. *GPU does not support Supervisor mode contexts.*
0	P: Present	PDP Entry is present. It must be "1" to point to a page directory pointer table

* HAW = 39 for client, and 46 for server.

PDPE for 1GB Page



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):30	ADDR: Address	Physical address of 1GB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
29:12	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
11	Local Memory	Physical Page is located in Local Memory instead of System Memory. Only applicable for device configurations with local device memory that is managed by the Device Driver instead of the OS. For other configurations it is ignored
10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	This bit must be "1" to point to a valid Page.

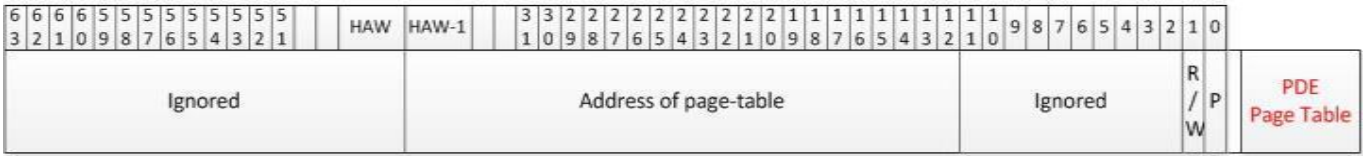
* HAW = 39 for client, and 46 for server.

PD: Pointer to Page Table

This section describes the following:

- PDE for Page Table
- PDE for 2 MB Page

PDE for Page Table



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page- table referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	PDP Entry is present. The value must be "1" to point to a page directory pointer table.

* HAW = 39 for client, and 46 for server.

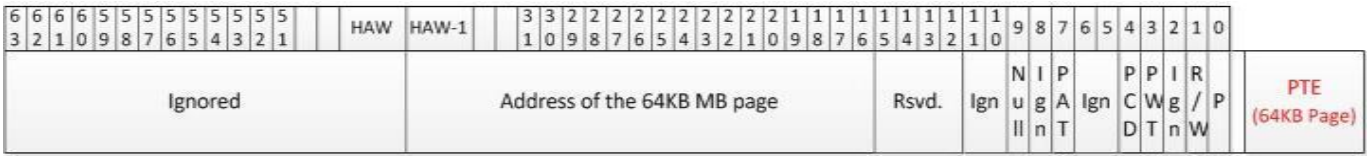
PDE for 2MB Page

Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):21	ADDR: Address	Physical address of 1GB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
20:12	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
11	Local Memory	Physical Page is located in Local Memory instead of System Memory. Only applicable for device configurations with local device memory that is managed by the Device Driver instead of the OS. For other configurations it is ignored
10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant*

Bits	Field	Description
		extended-context-entry) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 1GB Page.

* HAW = 39 for client, and 46 for server.

PTE: Page Table Entry for 64KB Page

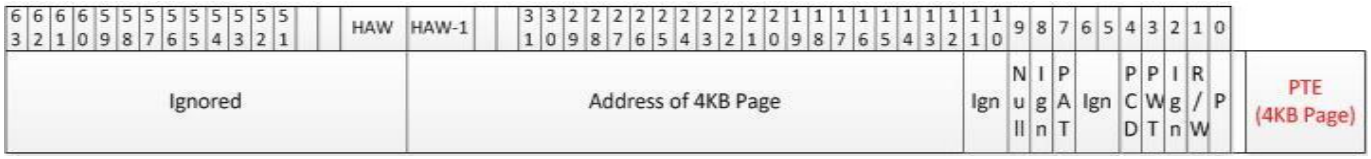


Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):16	ADDR: Address	Physical address of 64KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
15:12	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
11	Local Memory	Physical Page is located in Local Memory instead of System Memory. Only applicable for device configurations with local device memory that is managed by the Device Driver instead of the OS.
10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table

Bits	Field	Description
		referenced by this entry.
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (*and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry*) to the memory region controlled by this entry. See a later section for access rights. *GPU does not support Supervisor mode contexts.*
0	P: Present	It must be "1" to point to a 64KB Page.

* HAW = 39 for client, and 46 for server.

PTE: Page Table Entry for 4KB Page



Bits	Field	Description
63:HAW*	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
(HAW-1):12	ADDR: Address	Physical address of 64KB memory page referenced by this entry. This field is treated as Guest Physical Address (GPA) when Nested translations are enabled (NESTE=1) in the relevant extended-context entry.
11:10	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
9	N: Null	For Tile-Resources, private PPGTT tables enables for driver to merge Null Page information to primary (1st Level) translation tables. If Null=1, the h/w will avoid the memory access and return all zero's for the read access with a null completion, write accesses are dropped.
8	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.
6:5	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
4	PCD: Page level cache disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory-pointer table referenced by this entry.

Bits	Field	Description
3	PWT: Page level Write-through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the page directory- pointer table referenced by this entry.
2	*Ignored*	*Ignored (h/w does not care about values behind ignored registers)*
1	R/W: Read/Write	Write permission rights. If 0, write permission not granted for requests with user-level privilege (and requests with supervisor-level privilege, if WPE=1 in the relevant extended-context-entry) to the memory region controlled by this entry. See a later section for access rights. GPU does not support Supervisor mode contexts.
0	P: Present	It must be "1" to point to a 4KB Page.

* HAW = 39 for client, and 46 for server.

LNCFCMOSx

LNCFCMOCS0 - LNCF MOCS Register 0	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
Address:	0B020h

Programming Notes
WAReprogramMOCS: Upon render reset the driver needs to reprogram the LNCF MOCS Register.

DWord	Bit	Description
0	31	**Upper MOCS Index Mask Bit** Access: WO In order to prevent overwriting the upper MOCS index of this register, this bit must be set as part of the write.
	30:24	**Reserved** Access: RO Format: MBZ
	23:22	**Reserved** Access: RO Format: MBZ
	21:20	**Upper MOCS Index - L3 Cacheability Control** Access: R/W Lock Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index. For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	UPPER_DIRECT	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	UPPER_UC **[Default]**	Uncacheable
2h	UPPER_RESERVED	Reserved
3h	UPPER_WB	Writeback

DWord	Bit	Description
	19:17	**Upper MOCS Index - Skip Caching Control** Default Value: 0h Access: R/W Lock Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.

LNCFCMOCS0 - LNCF MOCS Register 0

Programming Notes

If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care.

If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.

Bit Offset	Corresponding Address Bit
0	10 ^ 16
1	11 ^ 17
2	12 ^ 18

16 **Upper MOCS Index - Enable Skip Caching**

Access:	R/W Lock

Enable the skip cache mechanism

Value	Name
0h	UPPER_ESC_DISABLE **[Default]**
1h	UPPER_ESC_ENABLE

15 **Lower MOCS Index Mask Bit**

Access:	WO

In order to prevent overwriting the lower MOCS index of this register, this bit must be set as part of the write.

14:8 **Reserved**

Access:	RO
Format:	MBZ

7:6 **Reserved**

Access:	RO
Format:	MBZ

5:4 **Lower MOCS Index - L3 Cacheability Control**

Access:	R/W Lock

Memory type information used in L3. This field is combined with the additional two bits that are sent by HDC based on binding table index.

For all other L3 requesters, this field is the primary source of L3 cache controls

Value	Name	Description
0h	LOWER_DIRECT **[Default]**	Use binding table index for direct EU accesses - for the rest it is reserved.
1h	LOWER_UC	Uncacheable
2h	LOWER_RESERVED	Reserved

LNCFCMOCS0 - LNCF MOCS Register 0		
	3h	LOWER_WB	Writeback
3:1	**Lower MOCS Index - Skip Caching Control**	
	Default Value:	0h
	Access:	R/W Lock
	Defines the bit values to enable caching. Outcome overrides the L3/LLC caching for the surface.	
	Programming Notes	
	If a given bit is programmed to 0, then the corresponding address bit value is treated as a don't care. If a given bit is programmed to 1, then the corresponding address bit must be 0 to cache in the target.	
	Bit Offset	**Corresponding Address Bit**
	0	10 ^ 16
	1	11 ^ 17
	2	12 ^18
0	**Lower MOCS Index - Enable Skip Caching**	
	Access:	R/W Lock
	Enable the skip cache mechanism	
	Value	**Name**
	0h	LOWER_ESC_DISABLE **[Default]**
	1h	LOWER_ESC_ENABLE

GLOB_MOCS_LECC_x

GLOB_MOCS_LECC_00_TC_00 - Global MOCS LECC 00 TC 00 Register	
Register Space:	MMIO: 0/2/0
Size (in bits):	32
_Custom_GTIReset:	DEV
_Custom_GTIIsContextSaved:	true
Address:	04000h
Name:	Global MOCS 0
ShortName:	GLOB_MOCS_0
Address:	04040h
Name:	Global MOCS 16
ShortName:	GLOB_MOCS_16
Address:	04080h
Name:	Global MOCS 32
ShortName:	GLOB_MOCS_32
Address:	040C0h
Name:	Global MOCS 48
ShortName:	GLOB_MOCS_48
MOCS register	

DWord	Bit	Description
0	31:19	**Reserved** Access: RO Format: MBZ
	18:17	**Self Snoop Enable** Access: R/W 00: Default value. Self snoop attribute sent to the uncore is as today - determined by MIDI unit logic 01: Override the self snoop bit generated by MIDI with 0. No self snoops are sent to the uncore for any transactions from this surface 11: Override the self snoop bit generated by MIDI with 1. Self snoops are always sent to the uncore for any transactions from this surface

Value	Name	Description
11b	Generated by MIDI with 1	Override the self snoop bit generated by MIDI with 1. Self snoops are always sent to the uncore for any transactions from this surface
01b	Generated by MIDI with 0	Override the self snoop bit generated by MIDI with 0. No self snoops are sent to the uncore for any transactions from this surface.

GLOB_MOCS_LECC_00_TC_00 - Global MOCS LECC 00 TC 00 Register		
	00b	Determined by MIDI unit logic **[Default]**	Default value. Self snoop attribute sent to the uncore is as today - determined by MIDI unit logic

16:15 **Class of Service**

Access:	R/W

Description
Class of Service sent to LLC to determine subset of ways the memory object will be stored in. 00: Class 0 01: Class 1 10: Class 2 11: Class 3 *Max* QoS:* Class 0 *Relative* QoS:* 0 > 1 > 2 ≥ 3** * Max/Relative statements above based on default/non-firmware-overridden GT QoS masks. ** CLOS2 = CLOS3 equivalence only on 4-way LLC SKUs.

Value	Name
00b	Class 0 **[Default]**
01b	Class 1
10b	Class 2
11b	Class 3

14 **Snoop Control Field**

Access:	R/W

Enables s/w to have GFX h/w to be able to consume IA generated buffers that are tagged as WB. Driver can mark these buffers as WB when generating them from IA

In SOCs, the fabric is not forced to be coherent all the time. IA-core generated WB buffers can only be consumed by GPU if that buffer is tagged as snoop-able in GPUs buffer definitions (or via GPU Page tables)

1: Non-Coherent Write/Read

0: Coherent Access

Note: There is a performance and power penalty in accessing surfaces that are tagged as snooped

Value	Name
0b	Coherent Access **[Default]**

	1b	Non-Coherent Write/Read

13:11 Page Faulting Mode

Default Value:	000b Global page faulting mode
Access:	R/W

This fields controls the page faulting mode that will be used in the memory interface block for the given request coming from this surface:

000: Use the global page faulting mode from context descriptor (default)

001-111: Reserved

10:8 Skip Caching control

Default Value:	000b Value is do not care
Access:	R/W
Format:	Enable[3]

Defines the bit values to enable caching. Outcome overrides the LLC caching for the surface.

If "0" - than corresponding address bit value is do not care

Bit[8]=1: address bit[9] needs to be "0" to cache in target

Bit[9]=1: address bit[10] needs to be "0" to cache in target

Bit[10]=1: address bit[11] needs to be "0" to cache in target

7 Enable Reverse Skip Caching

Access:	R/W
Format:	Enable

Enable for the Skip cache mechanism

0: Not enabled

1: Enabled for LLC

Value	Name
0b	Not Enabled **[Default]**
1b	Enabled for LLC

6 Dont allocate on miss

Access:	R/W

Controls defined for RO surfaces in mind, where if the target cache is missed - do not bring the line (applicable to LLC/eDRAM).

0: Allocate on MISS (normal cache behavior)

1: Do NOT allocate on MISS

Received confirmation from Altug on 03/13/13 that nothing needs to be done on this bit

GLOB_MOCS_LECC_00_TC_00 - Global MOCS LECC 00 TC 00 Register		
		Value / **Name** 0b — Allocate on MISS **[Default]** 1b — Not allocate on MISS
	5:4	**LRU management** Access: R/W **Description** This field allows the selection of AGE parameter for a given surface in LLC or eLLC. . If a particular allocation is done at youngest age ("3") it tends to stay longer in the cache as compared to older age allocations ("2", "1", or "0"). This option is given to driver to be able to decide which surfaces are more likely to generate HITs, hence need to be replaced least often in caches. LRU Age value is assigned as follows: 11: Assign the age of "3" 10: do not change the age on a hit. 01: Assign the age of "0" 00: Take the age value from Uncore CRs **Value** / **Name** 11b — Assign age 3 **[Default]** 10b — Do not change age 01b — Assign age 0 00b — Age value from Uncore
	3:2	**Target Cache** Access: R/W **Description** This field allows the choice of LLC vs eLLC for caching 00: eLLC Only 01: LLC Only 10: LLC/eLLC Allowed 11: LLC/eLLC Allowed **Value** / **Name** 00b — eLLC Only **[Default]** 01b — LLC Only

GLOB_MOCS_LECC_00_TC_00 - Global MOCS LECC 00 TC 00 Register		
		10b: LLC/eLLc Allowed 11b: LLC/eLLc Allowed 2
	1:0	**LLC/eDRAM cacheability control** Access: R/W **Description** Memory type information used in LLC/eDRAM. 00: Uncacheable (UC) 01: Uncacheable (WC) 10: Writethrough (WT) 11: Writeback (WB) Note: Final memory type is based on memory type resolution using MOCS and Page Table memory types and legacy/advanced mode Note: Binding table index based memory typing cannot be used for LLC/eDRAM memory type. Instead page table based controls have to be used Note: In case of SVM (advanced context), LLC/eDRAM memory type is used based on the page table controls and cannot be managed via MOCS index **Value** / **Name** 00b: Uncacheable (UC) 01b: Uncacheable (WC) 10b: Writethrough (WT) 11b: Writeback (WB) **[Default]**