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 13: SW/HW System Interface

April 2021, Revision 1.0

Notices and Disclaimers

Table of Contents

SW/HW System Interface

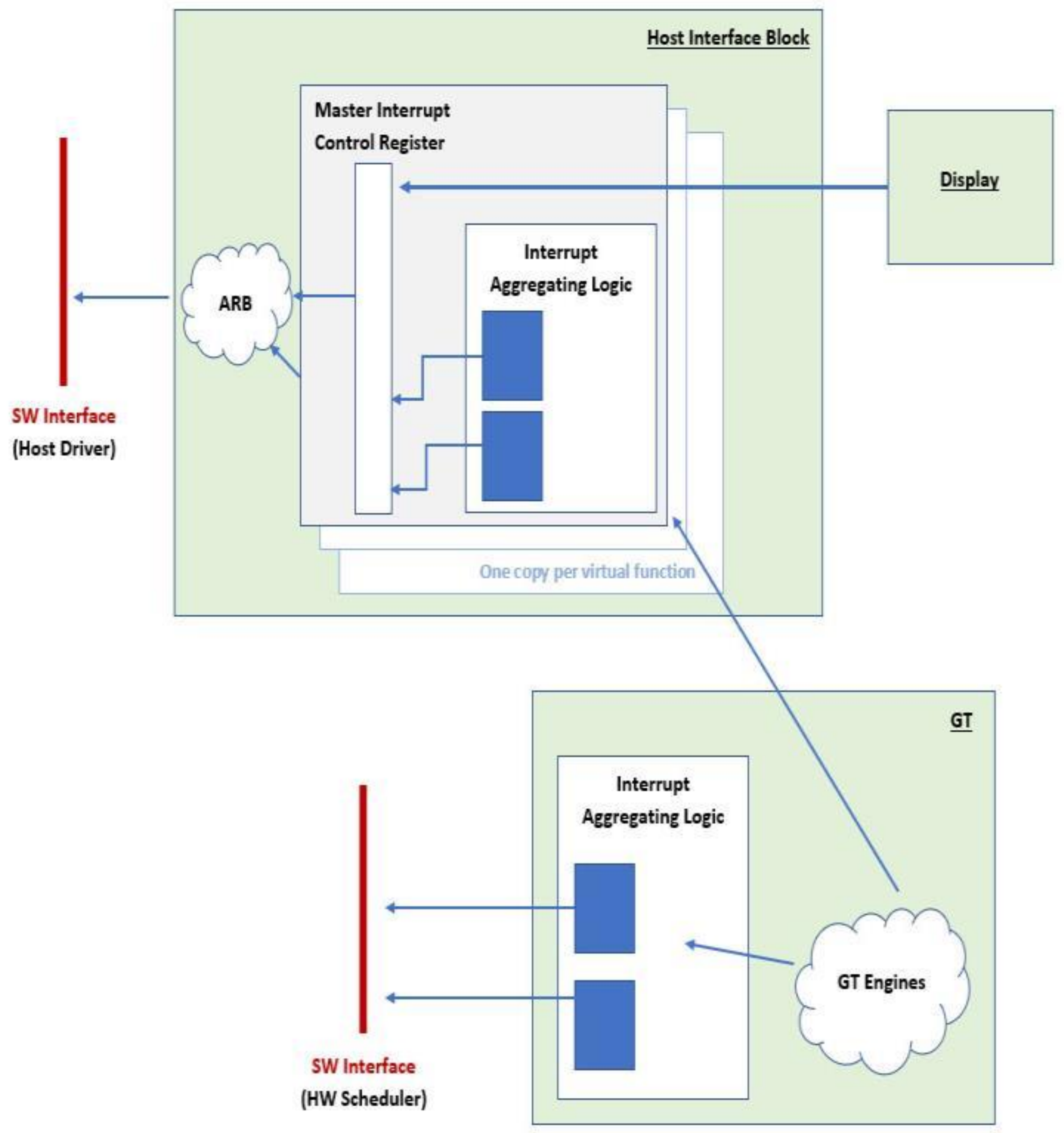
Interrupts

Overview:

The Graphics device is comprised of a number of independent engines that can be invoked to execute workloads. Engines communicate status primarily through interrupts. The Graphics device supports two models of scheduling and handling of interrupts:

- Host SW schedules and manages all interrupts
- Scheduling and related interrupts are managed by hardware scheduler (MinIA micro-controller) and host SW manages interrupts not related to scheduling.

The hardware can be configured to work in either of these models. HW scheduling is the preferred mode because it provides best utilization of resources. The figure below shows the high-level overview of the interrupt infrastructure.


Host Interface Block
Master Interrupt Control Register
Display
Interrupt Aggregating Logic
ARB
SW Interface
(Host Driver)
One copy per virtual function
GT
Interrupt Aggregating Logic
GT Engines
SW Interface
(HW Scheduler)

The interrupt infrastructure is designed to support both of these models. Each engine is allocated a set of interrupt bits that it can set to report events (the number of bits allotted to each engine varies -- most engines are allocated 16bits, some engines which have more events are allocated 32bits). Interrupt messages sent by engines result in interrupt bits being recorded in MMIO registers and an interrupt being generated to the servicing agent (MinIA scheduler or Host SW). The interrupt handler determines the source of the interrupt (by reading registers) and then processes the interrupts. Processing interrupts involves reading the interrupt status register, performing the operations for handling the interrupt and indicating completion of handling by writing to registers (clear).

When using the HW scheduler, the scheduling related interrupts are directed to the MinIA scheduler.

GT Engine Interrupts:

Within GT, engines are categorized into different engine classes and instances. An engine class is used to differentiate between engines that perform different functions (Copy, Render, VideoDecode, VideoEncode, etc). A product may have a number of instances of a specific engine class e.g.: GT2 has 2 instances of VD, GT3 has 4 instances of VD, etc. The following table lists various engine classes as well as instances within each class.

Engine Class	Engine Instance Name	ClassID[2:0]	InstanceID[5:0]
Render	RCS	0	0
Video Decode	VCS0-N	1	0-N
Video Enhancement Engine	VECS0-N/2	2	0-N/2
Copy Engine	BCS	3	0
Other			
	GTPM	4	1
	WDOAPerf	4	2
	SCTRG	4	3
	KCR	4	4
	Gunit	4	5
	CSME	4	6
Reserved		6-7	

Each engine reports up to 16 interrupts to interrupt handling logic. Source identification data is included in interrupt messages to interrupt aggregating logic, i.e. when reporting an interrupt to either host or graphics firmware, the generating engine must identify itself. 16 bits of identification is sent along with interrupt data, and comprises Engine Class ID, Instance ID and Virtual Function Number. Interrupt bit definition varies per engine class, these are listed in the Bspec in the Global/ section.

Format of interrupt message:

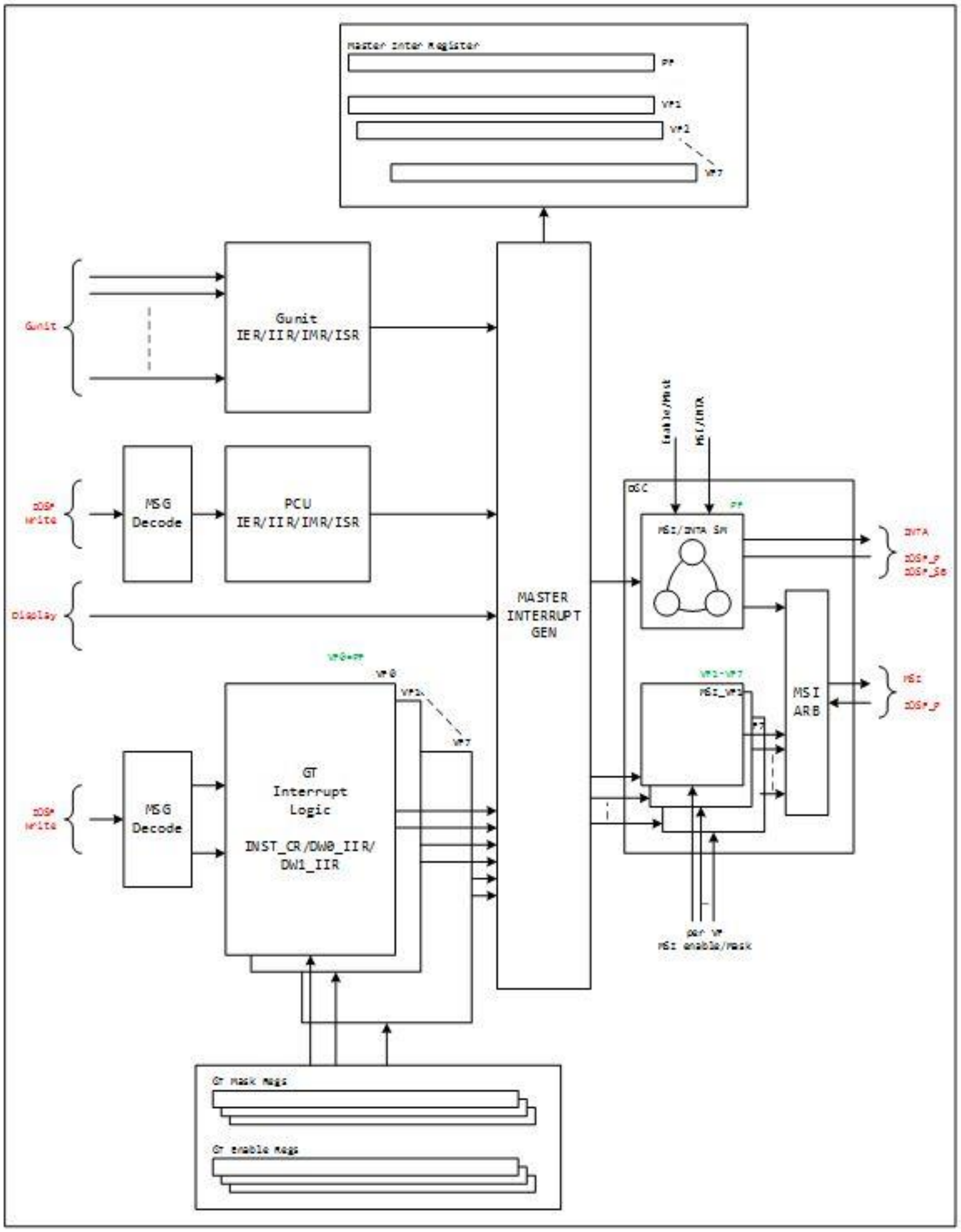
Bit Fied	Purpose
[31:30]	Reserved
[29:27]	VF ID
[26]	Reserved
[25:20]	Instance ID
[19]	Reserved
[18:16]	Engine Class ID
[15:0]	Interrupt data

Hardware Scheduler/MinIA SW Interface

Graphics interrupts to scheduling firmware are delivered as two unique vector values. Each vector accounts for 32 graphics engines. Firmware processes each of two groups of graphics engines independently.

Service routines are independent for the two interrupt vectors presented to the MinIA firmware.

Host SW Interface



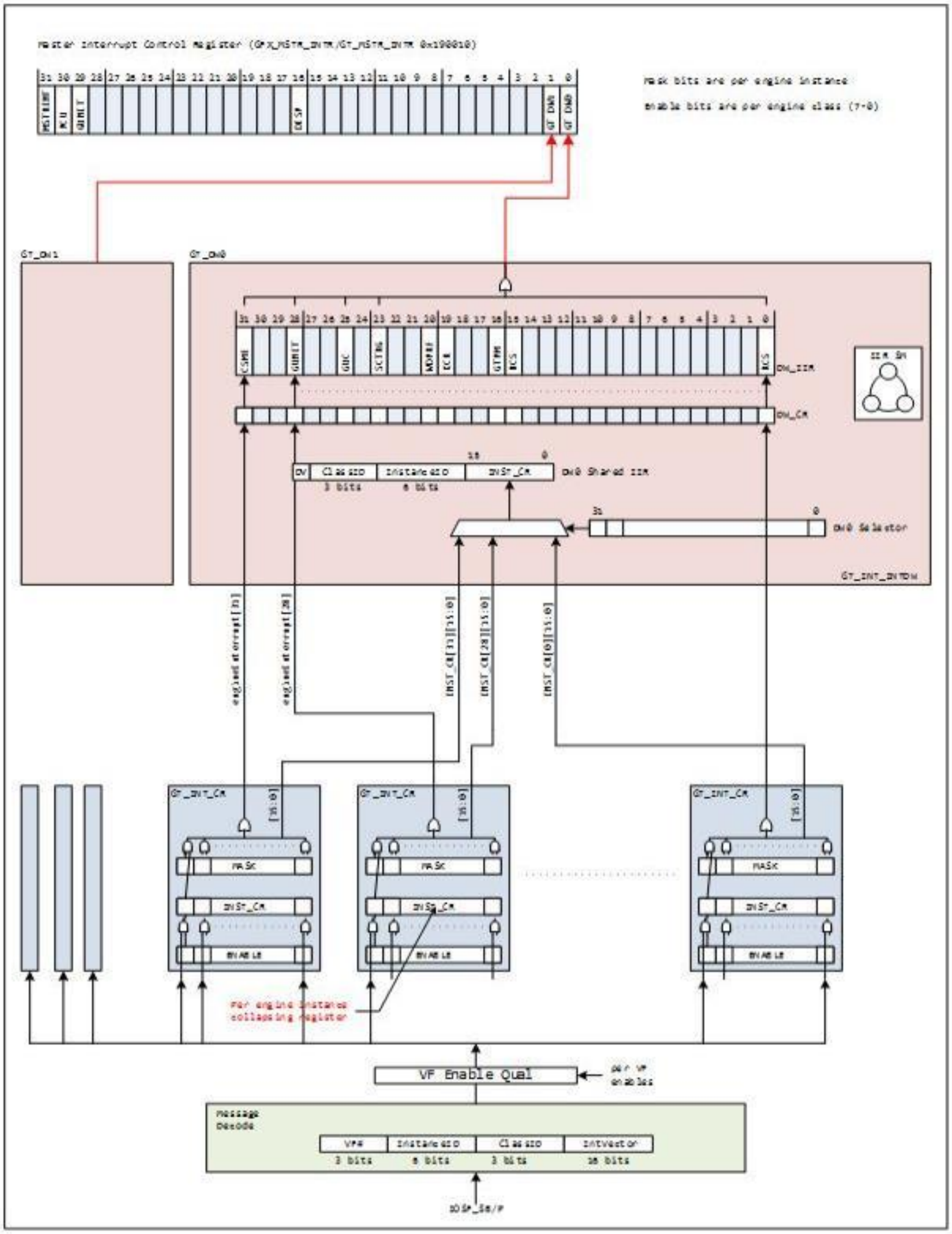
Interrupts to Host are delivered via a Master Interrupt Control Register. Graphics interrupts use 2 bits in the Master Interrupt Control Register. In addition, interrupt events from Display are also represented in the Master Interrupt Control Register. Multiple copies of INT DW and Master Interrupt Control Register exist, one for every virtual machine in the system.

Interrupt bits in the Master Interrupt Control Register are Read-Only bits, and are level indications that a second level interrupt is present (As seen earlier, second level interrupts per client are OR-ed together to set a bit in the Master. When the second level IIR is cleared, the bit represented in the Master will be 0.). An interrupt is sent to driver whenever bits are set in the Master Interrupt Control Register and the Enable bit is also set.

As a result of this interrupt, SW first resets the Master Control Enable bit. SW then reads the Master Interrupt Control register into a local variable, and works off this local variable to service interrupts. Once all lower level interrupts have been serviced, SW writes the Master Interrupt Control register to set the Master Control Enable bit.

Interrupt Aggregating Logic

A hierarchical interrupt status infrastructure is provided to efficiently determine the source of the interrupt. The first level of interrupts is generated by GT Engines. Interrupt handling logic accumulates these interrupts from the various engines, and organizes it as a single bit per engine in a second level. 32 bits of second level interrupts are OR-ed together to generate a DW-level interrupt event for up to 32 engines. Two such events are used to provide support for up to 64 GT engines. These DW-level interrupt events are marked in red in the picture below. When communicating with the MinIA, these events are mapped to two unique interrupt vectors in the MinIA LAPIC. When communicating with host driver, these events form two bits of the Master Interrupt Control Register as marked in the picture.


Master Interrupt Control Register (GFX_MSTR_INTR/GT_MSTR_INTR 0x190010)
MSTRINT
PCU
GUNIT
DISP
GT DW1
GT DW0
Mask bits are per engine instance
Enable bits are per engine class (7-0)
GT_DW1
GT_DW0
CSME
GUNIT
GUC
SCTRG
WDPERF
ECR
GTPM
BCS
RCS
DW_IIR
DW_CR
IIR SM
DV
Class ID
3 bits
Instance ID
6 bits
INST_CR
DW0 Shared IIR
DW0 Selector
GT_INT_INTDW
engine interrupt[31]
engine interrupt[28]
INST_CR[31][15:0]
INST_CR[28][15:0]
INST_CR[0][15:0]
GT_INT_CR
[15:0]
MASK
INST_CR
ENABLE
Per engine instance collapsing register
VF Enable Qual
per VF enables
Message Decode
VF#
3 bits
Instance ID
6 bits
Class ID
3 bits
IntVector
16 bits
IOSF_SB/P

Nomenclature:

First level interrupts are stored in storage named '**per-Instance Collapsing Register**' (INST_CR in the picture). These are 16 bits wide, one such storage exists per interrupt producing engine within GT.

Second level interrupts are stored in storage referred to as '**GT INT DW'** (DW_IIR, DW_CR in the picture). These are 32 bits wide. Two such DWs exist, DW0 and DW1

Each GT INT DW is accompanied by two other registers:

Selector: This is a one-hot version of GT INT DW, and indicates which engine of 32 is next to be serviced.

Shared IIR: This presents the 16 bit interrupt data of the engine selected by the Selector for SW to service.

First Level Interrupt Bits:

When an interrupt event comes into the interrupt handling logic, it is AND-ed with a per-Engine Enable register. Only enabled events make forward progress. Disabled events are simply dropped by the interrupt handling logic. [Note that multiple instances of the same engine type (except those in the 'Other' Engine Class) share the same Enable register.]

Enabled interrupts are logged in a per-instance Collapsing Register. These events are AND-ed with (the inverse of) a per-Instance Mask Register. Only unmasked events make forward progress. Masked events remain in the per-Instance Collapsing Register until they are unmasked. [Note that every instance (even of the same engine type) has its own Mask Register.]

Unmasked events in the per-Instance Collapsing Register are OR-ed together to produce a single second level interrupt event.

Second Level Interrupt Bits:

Second level interrupt events are stored in the GT INT DW. GT INT DW is a double buffered structure. A snapshot of events is taken when SW reads GT INT DW. From the time of read to the time of SW completely clearing GT INT DW (to indicate end of service), all incoming interrupts are logged in a secondary storage structure. This guarantees that the record of interrupts SW is servicing will not change while under service.

Bits in the GT INT DW_IIR are OR-ed together to generate a DW-level event. INT DW is cleared by writing 1s. If events exist in the secondary storage (DW_CR) at the time that INT DW is completely cleared, a second DW-level event will be generated.

Shared IIR, Selector:

The Shared IIR and Selector registers are used when SW is in the process of handling reported interrupts. As a result of a GT interrupt (DW-level interrupt), SW reads the DW IIR register. The read provides an indication of engines needing service. SW must then service engines one at a time by writing a one-hot selection into the Selector Register.

When a selection is made by writing the Selector, interrupt handling logic presents all the unmasked interrupt bits (first level interrupt events) for the selected engine in the Shared IIR, and sets the Data-Valid bit (MSB). SW can then read the Shared IIR and take action for the reported events. SW must clear the Shared IIR by writing 1 to the Data-Valid bit to indicate end of service for the selected engine. This clearing of the Shared IIR Data-Valid bit clears both the Shared IIR as well as the Selector. Note that the Selector data must be one-hot. Selector must not have a bit set that is not set in DW_IIR.

SW then repeats the above steps for each bit set in GT INT DW. Multiple rounds of Selector write-Shared IIR clear may be required to service GT INT DW a single time.

GT INT DW bits are cleared by SW writing a 1 to these bits. This is done only after individual engines are serviced via the Selector write –Shared IIR clear routine. This clearing can be done after each iteration through the Selector write-Shared IIR clear routine (i.e. one bit in GT INT DW cleared after each iteration), or all at once after all engines have been serviced. DW_IIR bits must not be cleared without first servicing that engine's interrupts via the Selector and Shared IIR registers.

Enable and Mask Registers:

Interrupt aggregating logic includes Enable registers per Engine Class. Different instances of the same engine class use the same Enable register, except for engines in the 'Other' class. Each instance in the 'Other' class has its own Enable register.

Interrupt aggregating logic also includes Mask registers. Each engine instance, even within the same Engine Class, has a unique Mask Register.

Enables for Engine classes at the two software interfaces are typically complements of each other.

External Interfaces

Uncore Registers

GFX MMIO – MCHBAR Aperture

Address:	140000h – 147FFFh 140000h - 14FFFFh
Default Value:	Same as MCHBAR
Access:	Aligned Word, Dword, or Qword Read

This range defined in the graphics MMIO range is an alias with which graphics driver can read and write registers defined in the MCHBAR MMIO space claimed through Device #0. Attributes for registers defined within the MCHBAR space are preserved when the same registers are accessed via this space. Registers that the graphics driver requires access to are Rank Throttling, GMCH Throttling, Thermal Sensor, etc.

The Alias functions work for MMIO access from the CPU, reads only. In addition, GT is also able to read registers within this range. Writes to the MCHBAR alias are not allowed (writes will have no affect).

Graphics MMIO registers can be accessed through MMIO BARs in the Gfx Device. The aliasing mechanism is turned off if memory access to the corresponding function is turned off via software or in certain power states.

Please refer to applicable EDS documentation for details of this register's format and behavior.

PCI Device Registers

Address	Symbol	Name
00054h	DEVEN_0_0_0_PCI	Device Enable
0005Ch	DPR_0_0_0_PCI	DMA Protected Range
00000h	VID2_0_2_0_PCI	Vendor Identification
00002h	DID2_0_2_0_PCI	Device Identification
00004h	PCICMD_0_2_0_PCI	PCI Command
00006h	PCISTS2_0_2_0_PCI	PCI Status
0000Ch	CLS_0_2_0_PCI	Cache Line Size
0000Dh	MLT2_0_2_0_PCI	Master Latency Timer
0000Eh	HDR2_0_2_0_PCI	Header Type
0000Fh	BIST_0_2_0_PCI	Built In Self Test
00010h	GTTMMADR_0_2_0_PCI	Graphics Translation Table Memory Mapped Range Address
00018h	GMADR_0_2_0_PCI	Graphics Memory Range Address
00020h	IOBAR_0_2_0_PCI	I/O Base Address
0002Ch	SVID2_0_2_0_PCI	Subsystem Vendor Identification
0002Eh	SID2_0_2_0_PCI	Subsystem Identification
00030h	ROMADR_0_2_0_PCI	Video BIOS ROM Base Address
00034h	CAPPOINT_0_2_0_PCI	Capabilities Pointer
0003Ch	INTRLINE_0_2_0_PCI	Interrupt Line
0003Dh	INTRPIN_0_2_0_PCI	Interrupt Pin
0003Eh	MINGNT_0_2_0_PCI	Minimum Grant
0003Fh	MAXLAT_0_2_0_PCI	Maximum Latency
00040h	CAPID0_0_2_0_PCI	Capability Identifier
00042h	CAPCTRL0_0_2_0_PCI	Capabilities Control
00044h	CAPID0_A_0_2_0_PCI	Capabilities A
00048h	CAPID0_B_0_2_0_PCI	Capabilities B
00050h	MGGC0_0_2_0_PCI	PCI Mirror of GMCH Graphics Control
00054h	DEVEN0_0_2_0_PCI	Mirror of Device Enable
00058h	DEV2CTL_0_2_0_PCI	Device 2 Control
00060h	MSAC_0_2_0_PCI	Multi Size Aperture Control
00070h	PCIECAPHDR_0_2_0_PCI	PCI Express Capability Header
00072h	PCIECAP_0_2_0_PCI	PCI Express Capability
00074h	DEVICECAP_0_2_0_PCI	Device Capabilities
0007Ah	DEVICESTS_0_2_0_PCI	PCI Express Capability Structure
000ACh	MSI_CAPID_0_2_0_PCI	Message Signaled Interrupts Capability ID

Address	Symbol	Name
000AEh	MC_0_2_0_PCI	Message Control
000B0h	MA_0_2_0_PCI	Message Address
000B4h	MD_0_2_0_PCI	Message Data
000B8h	MSI_MASK_0_2_0_PCI	MSI Mask Bits
000BCh	MSI_PEND_0_2_0_PCI	MSI Pending Bits
000C0h	BDSM_0_2_0_PCI	Mirror of Base Data of Stolen Memory
000C8h	GFX_VTDBAR_LSB_0_2_0_PCI	GFX_VTDBAR_LSB
000CCh	GFX_VTDBAR_MSB_0_2_0_PCI	
000D0h	PMCAPID_0_2_0_PCI	Power Management Capabilities ID
000D2h	PMCAP_0_2_0_PCI	Power Management Capabilities
000D4h	PMCS_0_2_0_PCI	Power Management Control and Status
000E0h	SWSMI_0_2_0_PCI	Software SMI
000E4h	GSE_0_2_0_PCI	Graphics System Event
000E8h	SWSCI_0_2_0_PCI	Software SCI
000FCh	ASLS_0_2_0_PCI	ASL Storage
00100h	PASID_EXTCAP_0_2_0_PCI	PASID Extended Capability Header
00104h	PASID_CAP_0_2_0_PCI	PASID Capability
00106h	PASID_CTRL_0_2_0_PCI	PASID Control
00200h	ATS_EXTCAP_0_2_0_PCI	ATS Extended Capability Header
00204h	ATS_CAP_0_2_0_PCI	ATS Capability
00206h	ATS_CTRL_0_2_0_PCI	ATS Control
00300h	PR_EXTCAP_0_2_0_PCI	Page Request Extended Capability Header
00304h	PR_CTRL_0_2_0_PCI	Page Request Control
00306h	PR_STATUS_0_2_0_PCI	Page Request Status
00308h	OPRC_0_2_0_PCI	Outstanding Page Request Capacity
0030Ch	OPRA_0_2_0_PCI	Outstanding Page Request Allocation
00320h	SRIOV_ECAPHDR_0_2_0_PCI	SRIOV Extended Capability Header
00324h	SRIOV_CAP_0_2_0_PCI	SRIOV Capabilities
00328h	SRIOV_CTRL_0_2_0_PCI	SRIOV Control Register
0032Ah	SRIOV_STS_0_2_0_PCI	SRIOV Status
0032Ch	SRIOV_INITVFS_0_2_0_PCI	SRIOV Initial VFs
0032Eh	SRIOV_TOTVFS_0_2_0_PCI	SRIOV Total VFs
00330h	SRIOV_NUMOFVFS_0_2_0_PCI	SRIOV Number of VFs
00334h	FIRST_VF_OFFSET_0_2_0_PCI	SRIOV First VF Offset
00336h	VF_STRIDE_0_2_0_PCI	VF Stride
0033Ah	VF_DEVICEID_0_2_0_PCI	VF Device ID

Address	Symbol	Name
0033Ch	SUPPORTED_PAGE_SIZES_0_2_0_PCI	**Supported Page Sizes**
00340h	SYSTEM_PAGE_SIZES_0_2_0_PCI	**System Page Sizes**
00344h	VF_BAR0_LDW_0_2_0_PCI	**VF BAR0 LDW**
00348h	VF_BAR0_UDW_0_2_0_PCI	**VF BAR0 UDW**
0034Ch	VF_BAR1_LDW_0_2_0_PCI	**VF BAR1 LDW**
00350h	VF_BAR1_UDW_0_2_0_PCI	**VF BAR1 UDW**
00354h	VF_BAR2_LDW_0_2_0_PCI	**VF BAR2 LDW**
00358h	VF_BAR2_UDW_0_2_0_PCI	**VF BAR2 UDW**
0035Ch	VF_MIGST_OFFSET_0_2_0_PCI	**VF Migration State Array Offset**

MMIO

Force Wake and Steering Table

Here is the Force Wake and Steering Table.

MMIO Range Start	MMIO Range End	# Bytes	Section	Target / Unit	Unit Description	Wake Target	Replicated / Multicast?	Replication Group Type	Inst. Count	Steering
00000000	00000AFF	2816		<reserved>	Reserved					
00000B00	00000BFF	256	gtgti	SF	Snoop filter	INF / COH	No	-	1	-
00000C00	00000DFF	512	gtgti	RPM	Ring PM unit	INF / COH	No	-	1	-
00000E00	00000FFF	512	gtgti	MGSR	Shadow Register and Interface to SA message channel	INF / COH	No	-	1	-
00001000	00001FFF	4096	gtgti	MDRB	Door Bell	INF / COH	No	-	1	-
00002000	000026FF	1792	gtfix	CS	Command Streamer	RENDER	No	-	1	-
00002700	00002FFF	2304	gtfix	OA	OAunit	GTI	No	-	1	-
00003000	00003FFF	4096	gtfix	CS	messaging range (3D Command Stream)	RENDER	No	-	1	-
00004000	000049FF	2560	gtgti	GAMW	Graphics Arbiter (GAM) Walk	GTI	No	-	1	-
00004A00	00004FFF	1536	gtgti	GAMT	Graphics Arbiter (GAM) TLB	GTI	No	-	1	-
00005000	000051FF	512		reserved						
00005200	000052FF	256	gtfix	SOL	SOL	RENDER	No	-	1	-
00005300	000053FF	256	gtfix	TSG0	TSG0	RENDER	No	-	1	-
00005400	000054FF	256	gtfix	TSG1	gtyscz					
00005500	00005FFF	2816	gtsc	WMBE	WMBE	RENDER	Yes	SLICE	2	sliceid[0..1]
00006000	00006FFF	4096	gtfix	SVG	SVG	RENDER	No	-	1	-
00007000	00007FFF	4096	gtsc	SVL	SVL	RENDER	Yes	SLICE	2	sliceid[0..1]

MMIO Range Start	MMIO Range End	# Bytes	Section	Target / Unit	Unit Description	Wake Target	Replicated / Multicast ?	Replication Group Type	Inst. Count	Steering
00008000	000080FF	256	gtgti	GPM	PM Unit Messages	GTI	No	-	1	-
00008100	0000813F	64	gtgti	GTGTICP	CP unit Messages	GTI	No	-	1	-
00008140	0000814F	16	gtsc	SCCFGCP	CP unit reg. file - Copy in Slice Common (in all slices)	RENDER	Yes	SLICE	2	sliceid[0..1]
00008150	0000815F	16	gtss	SSMCFGCP	CP unit reg. file - Copy in SSM	RENDER	Y (sub slice)			
00008160	000081FF	160		<reserved>	Reserved					
00008200	000082FF	256	gtgti	GDT	DTunit Messages	GTI	No	-	1	-
00008300	000084FF	512	gtfix	VF	VFunit Registers	RENDER	No	-	1	-
00008500	000085FF	256	gtgti	MBC	Boot Controller Messages	GTI	No	-	1	-
00008600	000086FF	256	gtgti	GPM	PM Unit Messages	GTI	No	-	1	-
00008700	000087FF	256	gtgti	MCFG	MCFG	GTI	No	-	1	-
00008800	0000883F	64	gtgti	DTF	Ring PM unit	INF / COH	No	-	1	-
00008840	00008BFF	960		<reserved>	Reserved					
00008C00	00008CFF	256	gtsc	SPM	PM Unit	RENDER	Yes	SLICE	2	sliceid[0..1]
00008D00	00008FFF	768		<reserved>	Reserved					
00009000	000093FF	1024	gtgti	MBC	BC unit	GTI	No	-	1	-
00009400	0000947F	128	gtgti	GTGTICP	CP unit reg. file - Copy in GTI par	GTI	No	-	1	-
00009480	000094CF	80		<reserved>	Reserved					
000094D0	0000951F	80	gtsc	SCCFGCP	CP unit reg. file - Copy in Slice Common (in all slices)	RENDER	Yes	SLICE	2	sliceid[0..1]
00009520	0000955F	64	gtssm	SSMCFGCP	CP unit reg. file - Copy in SSM	RENDER	Y (sub slice)			

MMIO Range Start	MMIO Range End	# Bytes	Section	Target / Unit	Unit Description	Wake Target	Replicated / Multicast ?	Replication Group Type	Inst. Count	Steering
00009560	000095FF	160	gtgti	INFCPCFG		INF / COH	No	-	1	-
00009600	000097FF	512		<reserved>	Reserved					
00009800	00009FFF	2048	gtgti	GDT	DT Unit (potentially some to OA)	GTI	No	-	1	-
0000A000	0000AFFF	4096	gtgti	GPM	PM Unit	GTI	No	-	1	-
0000B000	0000B0FF	256	gtsc	LNCF	L3 unique status registers for each slice (unicast per GT).	RENDER	Yes	SLICE	2	sliceid[0..1]
0000B100	0000B3FF	768	gtl3wrap	LBCF	L3 bank config space (multicast copy per bank and slice)	RENDER	Yes	L3BANK	2	sliceid[0..1]
0000B400	0000B47F	128	gtfix	LPFC	L3 performance / flush flow control unit	RENDER	No	-	1	-
0000B480	0000BFFF	2944		<reserved>	Reserved					
0000C800	0000CFFF	2048	gtgti	GAMW	range moved from GAMT to GAMW as per DCN 396119	GTI	No	-	1	-
0000D000	0000DEFF	3840	gtmc	<reserved>	Reserved					
0000DF00	0000DFFF	256	gtssm	CPSS	CPSS	RENDER	Yes	SUBSLICE	8	sliceid[0..1], subslice[0..3]
0000E000	0000E0FF	256	gtssm	DM	DM	RENDER	Yes	SUBSLICE	8	sliceid[0..1], subslice[0..3]
0000E100	0000E1FF	256	gtssm	SC	SC	RENDER	Yes	SUBSLICE	8	sliceid[0..1], subslice[0..3]
0000E200	0000E3FF	512	gtssm	GW	GWL (inst. 0)	RENDER	Yes	SUBSLICE	8	sliceid[0..1], subslice[0..3]

MMIO Range Start	MMIO Range End	# Bytes	Section	Target / Unit	Unit Description	Wake Target	Replicated / Multicast ?	Replication Group Type	Inst. Count	Steering
0000E400	0000E7FF	1024	gtssm	TDL	TDL	RENDER	Yes	SUBSLICE	8	sliceid[0..1], subslice[0..3]
0000E800	0000E8FF	256	gtssm	PSD	PSD	RENDER	Yes	SUBSLICE	8	sliceid[0..1], subslice[0..3]
0000E900	0000EAFF	512		<reserved>	Reserved					
0000EB00	0000EFFF	1280	gtgti	GAMWD	GAMWD	GTI	No	-	1	-
0000F000	000108FF	6400	gtgti	GAMW	Graphics Arbiter (GAM) Walk	GTI	No	-	1	-
00010900	00013FFF	14080		<reserved>	Reserved					
00014000	000143FF	1024	gtenc0	WIN	WIN unit	GTI	No	-	1	-
00014400	000147FF	1024	gtenc1	WIN	WIN unit	GTI	No	-	1	-
00014800	00016DFF	9728		<reserved>	Reserved					
00016E00	00016FFF	512	gtfix	VFR		RENDER	No	-	1	-
00017000	00017FFF	4096	gtfix	SVGR		RENDER	No	-	1	-
00018000	00019FFF	8192	gtfix	POCS		RENDER	No	-	1	-
0001A000	00021FFF	32768		<reserved>	Reserved					
00022000	00022FFF	4096	gtgti	BCS	Blitter Command Streamer	GTI	No	-	1	-
00023000	00023FFF	4096	gtgti	BCS	blitter messaging range (Blitter Command Stream)	GTI	No	-	1	-
00024000	0002407F	128	gtgti	GTISPC	GT SPC	INF / COH	No	-	1	-
00024080	000240FF	128	gtgti	MEDIASPC	WIN unit	GTI	No	-	1	-
00024100	0002417F	128	gtgti	MEDIA2SPC	WIN unit	GTI	No	-	1	-
00024180	000241FF	128	gtgti	SLICE0SPC	SLICE0 SPC unit	RENDER	No	-	1	-
00024200	0002427F	128	gtgti	NXTSLICESPC	SLICE1 SPC unit	RENDER	Yes	SLICE	2	sliceid[0..1]
00024280	000242FF	128	gtgti	SLICE1SPC <reserved>	Reserved <SLICE1 SPC unit)					

MMIO Range Start	MMIO Range End	# Bytes	Section	Target / Unit	Unit Description	Wake Target	Replicated / Multicast ?	Replication Group Type	Inst. Count	Steering
00024300	0002437F	128	gtgti	FIXSPC	fixed function SPC unit	GTI	No	-	1	-
00024380	000243FF	128		<reserved>	Reserved					
00024400	0002447F	128	gtsc	SSM0SPC	SSM0 SPC unit	RENDER	Yes	SLICE	2	sliceid[0..1]
00024480	000244FF	128	gtss	SSM1SPC	SSM1 SPC unit	RENDER	Yes	SLICE	2	sliceid[0..1]
00024500	0002457F	128	gtsc	SSM0SPC	SSM2 SPC unit	RENDER	Yes	SLICE	2	sliceid[0..1]
00024580	000245FF	128	gtss	SSM1SPC	SSM3 SPC unit	RENDER	Yes	SLICE	2	sliceid[0..1]
00024600	0002467F	128		<reserved>	Reserved					
00024680	000246FF	128	gtss	EUP1SPC	EU1SPC unit	RENDER	Yes	SUBSLICE	8	sliceid[0..1], subslice[0..3]
00024700	0002477F	128	gtss	EUP2SPC	EU2SPC unit	RENDER	Yes	SUBSLICE	8	sliceid[0..1], subslice[0..3]
00024780	000247FF	128	gtss	EUP3SPC	EU3SPC unit	RENDER	Yes	SUBSLICE	8	sliceid[0..1], subslice[0..3]
00024800	000249FF	512		<reserved>	Reserved					
00024A00	00024A7F	128	gtss	MSAMPSPC		RENDER	Yes	SUBSLICE	8	sliceid[0..1], subslice[0..3]
00024A80	00024DFF	896		<reserved>	Reserved					
00024E00	00024E7F	128	gtsc	SSM0SPC	SSM4 SPC unit	RENDER	Yes	SLICE	2	sliceid[0..1]
00024E80	00024EFF	128	gtss	SSM1SPC	SSM5 SPC unit	RENDER	Yes	SLICE	2	sliceid[0..1]
00024F00	00024F7F	128	gtsc	SSM0SPC	SSM6 SPC unit	RENDER	Yes	SLICE	2	sliceid[0..1]
00024F80	00024FFF	128	gtss	SSM1SPC	SSM7 SPC unit	RENDER	Yes	SLICE	2	sliceid[0..1]
00025000	000251FF	512		<reserved>						
00025200	0002527F	128	gtmc	MEDIASPC	EU3SPC unit	GTI	No	-	1	-
00025280	000252FF	128	gtmc	MEDIASPC	EU3SPC unit	GTI	No	-	1	-
00025300	0002FFFF	44288		<reserved>	Reserved					
00030000	0003FFFF	65536	gtgti	KCR		GTI	No	-	1	-

MMIO Range Start	MMIO Range End	# Bytes	Section	Target / Unit	Unit Description	Wake Target	Replicated / Multicast?	Replication Group Type	Inst. Count	Steering
001C0000	001C07FF	2048	gtyvdbox 0	VCS	Video Command Streamer	MEDIA0	No	-	1	-
001C0800	001C0FFF	2048	gtyvdbox 0	VIN	Media Units (VIN unit)	MEDIA0	No	-	1	-
001C1000	001C1FFF	4096	gtyvdbox 0	VCS	Video Command Streamer	MEDIA0	No	-	1	-
001C2000	001C27FF	2048	gtyvdbox 0	HUC	P24C micro-controller for HUC specific	MEDIA0	No	-	1	-
001C2800	001C2BFF	1024	gtyvdbox 0	HWM	HWM unit for HEVC	MEDIA0	No	-	1	-
001C2C00	001C3EFF	4864	gtyvdbox 0	<reserved>	Reserved					
001C3F00	001C3FFF	256	gtyvdbox 0	VDCFGCP	Media Units (VIN unit)	MEDIA0	No	-	1	-
001C4000	001C47FF	2048	gtyvdbox 1	VCS (Rsvd.)	Video Command Streamer					
001C4800	001C4FFF	2048	gtyvdbox 1	VIN (Rsvd.)	Media Units (VIN unit)					
001C5000	001C5FFF	4096	gtyvdbox 1	VCS (Rsvd.)	Video Command Streamer					
001C6000	001C67FF	2048	gtyvdbox 1	HUC (Rsvd.)	P24C micro-controller for HUC specific					
001C6800	001C6BFF	1024	gtyvdbox 1	HWM (Rsvd.)	HWM unit for HEVC					
001C6C00	001C7EFF	4864	gtyvdbox 1	<reserved>	Reserved					
001C7F00	001C7FFF	256	gtyvdbox 1	VDCFGCP (Rsvd.)	Media Units (VIN unit)					
001C8000	001C9FFF	8192	gtyvebox 0	VECS	Video Enhancement Command Streamer	MEDIA0	No	-	1	-
001CA000	001CA0FF	256	gtyvebox 0	VFW	VFW unit UPPER SLICE	MEDIA0	No	-	1	-
001CA100	001CBEFF	7680	gtyvebox 0	<reserved>	Reserved					
001CBF00	001CBFFF	256	gtyvebox	VECFGCP	Media Units	MEDIA0	No	-	1	-

MMIO Range Start	MMIO Range End	# Bytes	Section	Target / Unit	Unit Description	Wake Target	Replicated / Multicast?	Replication Group Type	Inst. Count	Steering
			0		(VIN unit)					
001CC000	001CFFFF	16384	RSVD	<reserved>	Reserved					
001D0000	001D07FF	2048	gtyvdbox 2	VCS	Video Command Streamer	MEDIA2	No	-	1	-
001D0800	001D0FFF	2048	gtyvdbox 2	VIN	Media Units (VIN unit)	MEDIA2	No	-	1	-
001D1000	001D1FFF	4096	gtyvdbox 2	VCS	Video Command Streamer	MEDIA2	No	-	1	-
001D2000	001D27FF	2048	gtyvdbox 2	HUC	P24C micro-controller for HUC specific	MEDIA2	No	-	1	-
001D2800	001D2BFF	1024	gtyvdbox 2	HWM	HWM unit for HEVC	MEDIA2	No	-	1	-
001D2C00	001D3EFF	4864	gtyvdbox 2	<reserved>	Reserved					
001D3F00	001D3FFF	256	gtyvdbox 2	VDCFGCP	Media Units (VIN unit)	MEDIA2	No	-	1	-
001D4000	001D47FF	2048	gtyvdbox 3	VCS (Rsvd.)	Video Command Streamer					
001D4800	001D4FFF	2048	gtyvdbox 3	VIN (Rsvd.)	Media Units (VIN unit)					
001D5000	001D5FFF	4096	gtyvdbox 3	VCS (Rsvd.)	Video Command Streamer					
001D6000	001D67FF	2048	gtyvdbox 3	HUC (Rsvd.)	P24C micro-controller for HUC specific					
001D6800	001D6BFF	1024	gtyvdbox 3	HWM (Rsvd.)	HWM unit for HEVC					
001D6C00	001D7EFF	4864	gtyvdbox 3	<reserved>	Reserved					
001D7F00	001D7FFF	256	gtyvdbox 3	VDCFGCP (Rsvd.)	Media Units (VIN unit)					
001D8000	001D9FFF	8192	gtyvebox 1	VECS (Rsvd.)	Video Enhancement Command Streamer					
001DA000	001DA0FF	256	gtyvebox 1	VFW (Rsvd.)	VFW unit UPPER SLICE					

MMIO Range Start	MMIO Range End	# Bytes	Section	Target / Unit	Unit Description	Wake Target	Replicated / Multicast ?	Replication Group Type	Inst. Count	Steering
001DA100	001DBEFF	7680	gtyvebox1	<reserved>	Reserved					
001DBF00	001DBFFF	256	gtyvebox1	VECFGCP (Rsvd.)	Media Units (VIN unit)					
001DC000	001DFFFF	16384	RSVD	<reserved>	Reserved					
001E0000	001E07FF	2048	gtyvdbox4	VCS (Rsvd.)	Video Command Streamer					
001E0800	001E0FFF	2048	gtyvdbox4	VIN (Rsvd.)	Media Units (VIN unit)					
001E1000	001E1FFF	4096	gtyvdbox4	VCS (Rsvd.)	Video Command Streamer					
001E2000	001E27FF	2048	gtyvdbox2	HUC (Rsvd.)	P24C micro-controller for HUC specific					
001E2800	001E2BFF	1024	gtyvdbox4	HWM (Rsvd.)	HWM unit for HEVC					
001E2C00	001E3EFF	4864	gtyvdbox4	<reserved>	Reserved					
001E3F00	001E3FFF	256	gtyvdbox4	VDCFGCP (Rsvd.)	Media Units (VIN unit)					
001E4000	001E47FF	2048	gtyvdbox5	VCS (Rsvd.)	Video Command Streamer					
001E4800	001E4FFF	2048	gtyvdbox5	VIN (Rsvd.)	Media Units (VIN unit)					
001E5000	001E5FFF	4096	gtyvdbox5	VCS (Rsvd.)	Video Command Streamer					
001E6000	001E67FF	2048	gtyvdbox5	HUC (Rsvd.)	P24C micro-controller for HUC specific					
001E6800	001E6BFF	1024	gtyvdbox5	HWM (Rsvd.)	HWM unit for HEVC					
001E6C00	001E7EFF	4864	gtyvdbox5	<reserved>	Reserved					
001E7F00	001E7FFF	256	gtyvdbox5	VDCFGCP (Rsvd.)	Media Units (VIN unit)					
001E8000	001E9FFF	8192	gtyvdbox5	VECS (Rsvd.)	Video Enhancement Command					

MMIO Range Start	MMIO Range End	# Bytes	Section	Target / Unit	Unit Description	Wake Target	Replicated / Multicast ?	Replication Group Type	Inst. Count	Steering
					Streamer					
001EA000	001EA0FF	256	gtyvebox 2	VFW (Rsvd.)	VFW unit UPPER SLICE					
001EA100	001EBEFF	7680	gtyvebox 2	<reserved>	Reserved					
001EBF00	001EBFFF	256	gtyvebox 2	VECFGCP (Rsvd.)	Media Units (VIN unit)					
001EC000	001EFFFF	16384	RSVD	<reserved>	Reserved					
001F0000	001F07FF	2048	gtyvdbox 6	VCS (Rsvd.)	Video Command Streamer					
001F0800	001F0FFF	2048	gtyvdbox 6	VIN (Rsvd.)	Media Units (VIN unit)					
001F1000	001F1FFF	4096	gtyvdbox 6	VCS (Rsvd.)	Video Command Streamer					
001F2000	001F27FF	2048	gtyvdbox 6	HUC (Rsvd.)	P24C micro-controller for HUC specific					
001F2800	001F2BFF	1024	gtyvdbox 6	HWM (Rsvd.)	HWM unit for HEVC					
001F2C00	001F3EFF	4864	gtyvdbox 6	<reserved>	Reserved					
001F3F00	001F3FFF	256	gtyvdbox 6	VDCFGCP (Rsvd.)	Media Units (VIN unit)					
001F4000	001F47FF	2048	gtyvdbox 7	VCS (Rsvd.)	Video Command Streamer					
001F4800	001F4FFF	2048	gtyvdbox 7	VIN (Rsvd.)	Media Units (VIN unit)					
001F5000	001F5FFF	4096	gtyvdbox 7	VCS (Rsvd.)	Video Command Streamer					
001F6000	001F67FF	2048	gtyvdbox 7	HUC (Rsvd.)	P24C micro-controller for HUC specific					
001F6800	001F6BFF	1024	gtyvdbox 7	HWM (Rsvd.)	HWM unit for HEVC					
001F6C00	001F7EFF	4864	gtyvdbox 7	<reserved>	HWM unit for HEVC					
001F7F00	001F7FFF	256	gtyvdbox 7	VDCFGCP (Rsvd.)	Media Units (VIN unit)					

MMIO Range Start	MMIO Range End	# Bytes	Section	Target / Unit	Unit Description	Wake Target	Replicated / Multicast?	Replication Group Type	Inst. Count	Steering
001F8000	001F9FFF	8192	gtyvebox 3	VECS (Rsvd.)	Video Enhancement Command Streamer					
001FA000	001FA0FF	256	gtyvebox 3	VFW (Rsvd.)	VFW unit UPPER SLICE					
001FA100	001FBEFF	7680	gtyvebox 3	<reserved>	Reserved					
001FBF00	001FBFFF	256	gtyvebox 3	VECFGCP (Rsvd.)	Media Units (VIN unit)					
001FC000	001FFFFF	16384	RSVD	<reserved>	Reserved					
00200000	0023FFFF	262144	RSVD	<reserved for media slices 4-7>	Reserved for media slices 4-7					

- The Steering Control Registers reside at the following locations:
  - MGSR access point (access initiated by agent outside of GT):

#	Steering Reg Addr	Description
4	0xFDC	Access steering towards all other endpoints (all but above)

- CS access point:

#	Steering Reg Addr	Description
1	0x20CC	Access steering towards all GT endpoints

- **Note:** All Steering Control Registers contain the following fields:

Field	Description
multicast	1: Access will be multicast to all replicated endpoints: • *WRITE* op cycles go to all endpoint instances; sliceid[]/subsliceid[] fields ignored. • *READ* op cycles go to all endpoint instances, and responses are returned from all instances; The MsgCh selects single instance's response as the final read return, based on sliceid[]/subsliceid[] fields. 0: Access will be steered using sliceid[] and subsliceid[] fields below: • Both *WRITE* and *READ* cycles go to a single instance of an endpoint,

	based on sliceid[]/subsliceid[] steering. Default: 1 Note: The multicast field has no impact for a non-replicated target.
sliceid[]	Default: 0
subsliceid[]	Default: 0

- The following Replication Group Types exist for multicast MMIO endpoints:

Replication Group Type	Description / Notes
SLICE	• All actual products are 1 slice for fuse/steering purposes
SS	• 8 subslices • Uses subsliceid field • Terminated/disabled when the corresponding subslice fuse is disabled
L3BANK	• 8 instances • subsliceid 0..7 to access • Terminated/disabled when fuse_gt_l3disable bit mapped to this bank is 'disable'

- Fuse reflections (how to tell when an endpoint is disabled):

Fuse	Register reflection
fuse_gt_ss_dis[4:0]	0x913C[7:0]
fuse_gt_l3_disable[3:0]	0x9118[7:0] (fuse is replicated into [3:0] and [7:4])

Note: MsgCh termination also occurs when the domains are powered down. (i.e., not necessarily because the domain is disabled/fused off.) If reading/writing the registers is needed, then force-wake of the domain is required. Force-wake is not required for shadow register accesses coming through MGSR.

- The following table captures the force-wake and corresponding acknowledgment register locations for the various domains:

Domain	Driver ForceWake Req	Driver ForceWake Ack	ForceWake Req	ForceWake Status	Comment
AON	NA	NA	NA	NA	Registers sit outside of the C6 boundary. No ForceWake required.

Domain	Driver ForceWake Req	Driver ForceWake Ack	ForceWake Req	ForceWake Status	Comment
GT	0xA188	0x00130044	NA	NA	
Render	0xA278	0x0D84	0xA27C	0xA2A0[1]	
VDBOX0	0xA540	0x0D50	0xA274[0]	0xA2A0[0]	
VDBOX1	0xA544	0x0D54	0xA274[1]	0xA2A0[0]	
VDBOX2	0xA548	0x0D58	0xA274[2]	0xA2A0[2]	
VDBOX3	0xA54C	0x0D5C	0xA274[3]	0xA2A0[2]	As available per config
VDBOX4	0xA550	0x0D60	0xA274[4]	0xA2A0[3]	
VDBOX5	0xA554	0x0D64	0xA274[5]	0xA2A0[3]	
VDBOX6	0xA558	0x0D68	0xA274[6]	0xA2A0[4]	
VDBOX7	0xA55C	0x0D6C	0xA274[7]	0xA2A0[4]	
VEBOX0	0xA560	0x0D70	0xA274[8]	0xA2A0[0]	
VEBOX1	0xA564	0x0D74	0xA274[9]	0xA2A0[2]	
VEBOX2	0xA568	0x0D78	0xA274[10]	0xA2A0[3]	
VEBOX3	0xA56C	0x0D7C	0xA274[11]	0xA2A0[4]	

- Miscellaneous Notes:
- The MsgCh network has termination points, where cycles to endpoints that are disabled (fused-off, powered off, etc...) are gracefully completed. The termination node on the network will sink P cycles, and return dummy completions for NP cycles, on behalf of the disabled endpoints.
- Access requirements to registers that are part of GTMMADDR but not listed in the GT MMIO map table is defined elsewhere. This descriptions in this document only cover GT range.

Multicast Steering and Die Recovery

Some units in GT are replicated multiple times in the design, each with their own register storage local to that instance.

- In some cases, each replica/instance gets its own MMIO address range of offsets – for example, the multiple CCS command streamers, multiple VDBox/VEBox instances. For those, direct register access targets the only instance of that registers. The programming model described on this page is moot for those cases where each register has unique address.
- In other cases, the multiple instances of the unit use the same MMIO address on message channel. For these cases, the message channel provides additional capabilities to address the instances for read/write operations in either multicast (targeting all instances) or unicast modes (target specific instance) via a set of "steering registers" which can be configured to direct the access as desired. The steering registers have 3 fields: Multicast/Unicast, Sliceid, Subsliceid.
    - Multicast write access - write goes to all instances; sliceid/subsliceid fields are ignored
    - Multicast read access – read goes to all instances and all instances generate read response; message channel selects single instance's response as the final read return (based on the steering register slice/subslice fields)
    - Unicast write access – write goes to only the instance specified in the steering register
    - Unicast read access – read goes to only the instance specified in the steering register
    - In some replicated units, all of the replicated instances always "enabled" from a message channel perspective (never fused off/separately power gated) and thus all instances are always accessible if the containing power well is on (e.g. if GT is out of RC6)
- In some replicated units, there are die recovery/fuse down modes where some instances are fused off/disabled. For the latter, GT also contains MMIO registers which allow SW to detect which instances are fused as enabled/disabled (generally 1-hot). When this fuse down case applies, message channel is aware of the fusing and provides automatic termination of cycles toward disabled instances (writes get dropped with dummy NP completion if NP write; reads get dummy completion with 0 read return value from that instance). The fuse mirror register provides a mechanism for SW to know which instances are valid and to program the steering register toward enabled instances when needed – see comments below.

General rules:

- Some of these replicated registers are control registers which are generally expected to be all programmed with the same value – for these, writes should generally be multicast and reads can target any enabled instance (since all instances should contain the same value from prior multicast write).
- Some replicated registers are status registers and are expected to have different values as part of normal usage (for example, INSTDONE registers related to Sampler, Slice common; TDL thread status, etc). For these typical usage model would be to either iterate over all enabled instances or select specific single instance to target.

- If an instance is disabled (access terminated on message channel via the fuse info above or if containing power well is power gated), reads from that instance will return 0s and writes are silently dropped. Since the default for the steering registers is multicast read with sliceid=subsliceid=0, the default hardware behavior is to return data from instance that corresponds with sliceid/subsliceid = 0. If that instance is disabled, message channel will return a dummy response (0). In order to get correct/valid value the steering registers must be used to access a valid instance.
    - Note that a common usage model is for SW/FW to initializing specific bits in control register by reading the current/default value, then modifying the value in memory (set/clear few bits), and then write the result back.
    - For these cases, SW must ensure that it uses the steering registers to steer to an enabled instance when performing the initial read.
- When performing engine and power context save restore, GT hardware is aware of the fuses and internally targets reads for context save toward the first enabled instance.
- In cases where steering registers are being programmed, caution must be exercised to ensure that there is no race condition/concurrent access between two different initiators using a given steering register. SW must protect against concurrent access by multiple threads to any given steering register. System level flows must also guard against concurrent access by Firmware (CSC/FSP FW, Punit pCode) and driver tools to any given steering register.
    - Multicast is the hardware default. If an agent sets a steering register to unicast mode, they should generally set it back to multicast after completion.
    - In some projects there are separate steering registers listed are intended to allow for some degree of concurrency between different usages targeting different destinations in GT by replication group.
        - MGSR uses the MMIO offset requested in the inbound cycle to select which steering register to use for routing.
        - MGSR uses SAI policy registers to identify sources as “IA” (low privilege cfg_src on message channel) vs “HW” (high privilege – includes trusted firmware such as CSC/FSP, Pcode)
        - See project specific documentation for the list of steering registers and their intended use.

SW Virtualization Reserved MMIO range

The MMIO address range from 0x178000 thru 0x178FFF is reserved for communication between a VMM and the GPU Driver executing on a Virtual Machine.

HW does not actually implement anything within this range. Instead, in a SW Virtualized environment, if a VM driver issues a read to this MMIO address range, the VMM will trap that access, and provide whatever data it wishes to pass to the VM driver. In a non-SW-Virtualizated environment (including an SR-IOV Virtualized environment), reads will return zeros, like any other unimplemented MMIO address. Writes to this range are always ignored.

It is important that no "real" HW MMIO register be defined within this range, as it would be inaccessable in a SW-virtualized environment.

Observability

Observability Overview

As GFX-enabled systems and usage models have grown in complexity over time, a number of hardware features have been added to provide more insight into hardware behavior while running a commercially available operating system. This chapter documents these features with pointers to relevant sections in other chapters. Supported observability features include:

Feature
Performance counters
Internal node tracing

Note: This chapter describes the registers and instructions used to monitor GPU performance. Please review other volumes in this specification to understand the terms, functionality and details for specific Intel graphics devices.

DFD Configuration Restore

Since DFD logic does not usually add value to end user usage models and its configuration space is large (which would add latency to power management restore flows), it is typically not enabled during normal operation for optimal power & performance. Hence, additional steps are required when DFD functionality is needed in combination with system configurations where GT logic loses power/is reset. The basic strategy per scenario is detailed below.

GT Power-up/RC6 Exit

Strategy
Replicate failure without power management

Render Engine Power-up

Configure the RCS RC6 W/A batch buffer to restore render engine DFD configuration ONLY.

Media Engine Power-up

Configure the applicable media command stream W/A batch buffer to restore media engine DFD configuration ONLY.

Resume from Partial GT Power Down

For cases where SW is aware of power well state, re-apply DFD configuration.

For cases where SW is not aware of power well state, configure the per-context W/A batch buffer to apply the DFD configuration on every context load.

Trace

This section contains the following contents:

Feature
• Performance Visibility

Performance Visibility

Motivation for Hardware-Assisted Performance Visibility

As the focus on GFX performance and programmability has increased over time, the need for hardware (HW) support to rapidly identify bottlenecks in HW and efficiently tune the work sent to same has become correspondingly important. This part of the BSpec describes the HW support for Performance Visibility.

Performance Event Counting

An earlier generation introduced dedicated GFX performance counters to address key issues associated with existing chipset CHAPs counters (lack of synchronization with GFX rendering work and low sampling frequency achievable when sampling via CPU MMIO read). Furthermore, reliance on SoC assets created a cross-IP dependency that was difficult to manage well. Hence, the approach since that earlier generation has been to use dedicated counters managed by the graphics device driver for graphics performance measurement. The dedicated counter values are written to memory whenever an MI_REPORT_PERF_COUNT command is placed in the ring buffer.

While this approach eliminated much of the error associated with the previous approaches, it is still limited to sampling the counters only at the boundaries between ring commands. This inherently limited the ability of performance analysis tools to drill down into a primitive, which can contain thousands of triangles and require several hundreds of milliseconds to render. It is further worth noting that precise sampling via MI_REPORT_PERF command requires flushing the GFX pipeline before and after the work of interest. The overhead of flushing the GFX pipeline can become large if the work of interest is small, hence reducing the accuracy of the performance counter measurement. In such situations, the flush can be removed or internally triggered reporting can be used with some resulting loss of precision in which draws/dispatches are being profiled.

Additionally, Intel design and architecture teams found that the existing silicon-based performance analysis tools provided only a general idea of where a problem may exist but were not able to pinpoint a problem. This was generally because the counter values are integrated across a very large time period, washing out the dynamic behavior of the workload.

Enhanced the aggregating counters to support the additional thread types generated by DX11 workloads. The high rate at which interesting internal events can occur motivated adding an interrupt-generation capability so that HW could notify SW when the data buffer was approaching full.

Enhances support for high reporting frequencies by increasing the report buffer size in order to allow SW sufficient time between performance monitoring interrupts, enabling single run histogramming support for events like pixels per polygon. Desire for more flexibility in custom event creation drove addition of 4 more Boolean counters.

Issues with support drove enhancements to enable performance monitoring with RC6 enabled, different report buffer ring wrap behavior, and MMIO visibility into performance counters.

Enhances functionality of aggregating counters for EUs by providing some flexibility in what quantities are aggregated across all EUs including more quantities relevant to GPGPU workloads. Since several of the previously defined aggregating counters had not delivered very much value on earlier projects, the overall number of A-counters has gone down even though aggregating counters for sampler/pixel-level functionality have been added/redefined. Custom counter creation has been enhanced by adding the ability to negate a signal at the input of the Boolean logic. Given that the increased complexity of GFX workloads and number of EUs in GT2/GT3 could lead to more frequent counter overflows, the width A-counters has increased to 40 bits. HW optimizations have also modified the SW interface slightly. Please note that no media-specific A-counter support was added, hence requiring all media engine support to be implemented using B/C-counters.

All OA config registers are tied to GT global reset and hence are not affected by per-engine resets (e.g. render only reset).

OA Programming Guidelines

SW utilizing OA HW is expected to monitor the overflow/lost report status for the OABUFFER and respond as appropriate for the active usage model.

In order for OA counters to increment the 'Counter Stop-Resume Mechanism' bit of the OACTXCONTROL register must be set. This requires a RCS context with this bit set be loaded, and either RCS force wake be enabled or the RCS context be left active for the duration of the window this counter is needed for.

In general, OA is effectively unable to count between the power context save that happens prior to GFX entering RC6 and the power context restore that occurs on the next RC6 exit. This limitation results from the fact that the counters themselves are power context save/restored and hence the counts that (may) have accumulated in this time window are overwritten by the saved values that are read back from the power context save area. An example of the kind of information that can be missed is the GTI traffic resulting from the power context save of OA itself. The size of this performance counting blind spot is microarchitecturally minimized as much as reasonably possible but still varies from device to device.

CPD must be disabled during performance counting or undercounts may occur.

HW Support

This section contains various reporting counters and registers for hardware support for Performance Visibility.

Performance Counter Registers

Register
OACTXCONTROL - Observation Architecture Control per Context
OACTXID - Observation Architecture Control Context ID
OA_IMR - OA Interrupt Mask Register
OASTATUS - Observation Architecture Status Register
OAHEADPTR - Observation Architecture Head Pointer
OATAILPTR - Observation Architecture Tail Pointer
OABUFFER - Observation Architecture Buffer
OASTARTTRIG_COUNTER - Observation Architecture Start Trigger Counter
OARPTTRIG_COUNTER - Observation Architecture Report Trigger Counter
OAREPORTTRIG2 - Observation Architecture Report Trigger 2
OAREPORTTRIG6 - Observation Architecture Report Trigger 6
CEC0-0 - Customizable Event Creation 0-0
CEC1-0 - Customizable Event Creation 1-0
CEC1-1 - Customizable Event Creation 1-1
CEC2-0 - Customizable Event Creation 2-0
CEC2-1 - Customizable Event Creation 2-1
CEC3-0 - Customizable Event Creation 3-0
CEC3-1 - Customizable Event Creation 3-1
CEC4-0 - Customizable Event Creation 4-0
CEC5-0 - Customizable Event Creation 5-0
CEC5-1 - Customizable Event Creation 5-1
CEC6-0 - Customizable Event Creation 6-0
CEC6-1 - Customizable Event Creation 6-1
CEC7-0 - Customizable Event Creation 7-0
CEC7-1 - Customizable Event Creation 7-1

The following Performance Statistics registers are power context save/restored:

Register
OAPERF_A0 - Aggregate Perf Counter A0
OAPERF_A0_UPPER - Aggregate Perf Counter A0 Upper DWord
OAPERF_A1 - Aggregate Perf Counter A1
OAPERF_A1_UPPER - Aggregate Perf Counter A1 Upper DWord
OAPERF_A2 - Aggregate Perf Counter A2
OAPERF_A2_UPPER - Aggregate Perf Counter A2 Upper DWord
OAPERF_A3 - Aggregate Perf Counter A3
OAPERF_A3_UPPER - Aggregate Perf Counter A3 Upper DWord
OAPERF_A4 - Aggregate Perf Counter A4
OAPERF_A4_UPPER - Aggregate Perf Counter A4 Upper DWord
OAPERF_A4_LOWER_FREE - Aggregate Perf Counter A4 Lower DWord Free

Register
OAPERF_A4_UPPER_FREE - Aggregate Perf Counter A4 Upper DWord Free
OAPERF_A5 - Aggregate Perf Counter A5
OAPERF_A5_UPPER - Aggregate Perf Counter A5 Upper DWord
OAPERF_A6 - Aggregate Perf Counter A6
OAPERF_A6_UPPER - Aggregate Perf Counter A6 Upper DWord
OAPERF_A6_LOWER_FREE - Aggregate Perf Counter A6 Lower DWord Free
OAPERF_A6_UPPER_FREE - Aggregate Perf Counter A6 Upper DWord Free
OAPERF_A7 - Aggregate Perf Counter A7
OAPERF_A7_- Upper Aggregate Perf Counter A7 Upper DWord
OAPERF_A8 - Aggregate Perf Counter A8
OAPERF_A8_UPPER - Aggregate Perf Counter A8 Upper DWord
OAPERF_A9 - Aggregate Perf Counter A9
OAPERF_A9_UPPER - Aggregate Perf Counter A9 Upper DWord
OAPERF_A10 - Aggregate Perf Counter A10
OAPERF_A10_UPPER - Aggregate Perf Counter A10 Upper DWord
OAPERF_A11 - Aggregate Perf Counter A11
OAPERF_A11_UPPER - Aggregate Perf Counter A11 Upper DWord
OAPERF_A12 - Aggregate Perf Counter A12
OAPERF_A12_UPPER - Aggregate Perf Counter A12 Upper DWord
OAPERF_A13 - Aggregate Perf Counter A13
OAPERF_A13_UPPER - Aggregate Perf Counter A13 Upper DWord
OAPERF_A14 - Aggregate Perf Counter A14
OAPERF_A14_UPPER - Aggregate Perf Counter A14 Upper DWord
OAPERF_A15 - Aggregate Perf Counter A15
OAPERF_A15_UPPER - Aggregate Perf Counter A15 Upper DWord
OAPERF_A16 - Aggregate Perf Counter A16
OAPERF_A16_UPPER - Aggregate Perf Counter A16 Upper DWord
OAPERF_A17 - Aggregate Perf Counter A17
OAPERF_A17_UPPER - Aggregate Perf Counter A17 Upper DWord
OAPERF_A18 - Aggregate Perf Counter A18
OAPERF_A18_UPPER - Aggregate Perf Counter A18 Upper DWord
OAPERF_A19 - Aggregate Perf Counter A19
OAPERF_A19_UPPER - Aggregate Perf Counter A19 Upper DWord
OAPERF_A19_LOWER_FREE - Aggregate Perf Counter A19 Lower DWord Free
OAPERF_A19_UPPER_FREE - Aggregate Perf Counter A19 Upper DWord Free
OAPERF_A20 - Aggregate Perf Counter A20
OAPERF_A20_UPPER - Aggregate Perf Counter A20 Upper DWord
OAPERF_A20_UPPER_FREE - Aggregate Perf Counter A20 Upper DWord Free
OAPERF_A20_LOWER_FREE - Aggregate Perf Counter A20 Lower DWord Free
OAPERF_A21 - Aggregate Perf Counter A21
OAPERF_A21_UPPER - Aggregate Perf Counter A21 Upper DWord

Register
OAPERF_A22 - Aggregate Perf Counter A22
OAPERF_A22_UPPER - Aggregate Perf Counter A22 Upper DWord
OAPERF_A23 - Aggregate Perf Counter A23
OAPERF_A23_UPPER - Aggregate Perf Counter A23 Upper DWord
OAPERF_A24 - Aggregate Perf Counter A24
OAPERF_A24_UPPER - Aggregate Perf Counter A24 Upper DWord
OAPERF_A25 - Aggregate Perf Counter A25
OAPERF_A25_UPPER - Aggregate Perf Counter A25 Upper DWord
OAPERF_A26 - Aggregate Perf Counter A26
OAPERF_A26_UPPER - Aggregate Perf Counter A26 Upper DWord
OAPERF_A27 - Aggregate Perf Counter A27
OAPERF_A27_UPPER - Aggregate Perf Counter A27 Upper DWord
OAPERF_A28 - Aggregate Perf Counter A28
OAPERF_A28_UPPER - Aggregate Perf Counter A28 Upper DWord
OAPERF_A29 - Aggregate Perf Counter A29
OAPERF_A29_UPPER - Aggregate Perf Counter A29 Upper DWord
OAPERF_A30 - Aggregate Perf Counter A30
OAPERF_A30_UPPER - Aggregate Perf Counter A30 Upper DWord
OAPERF_A31 - Aggregate_Perf_Counter_A31
OAPERF_A31_UPPER - Aggregate Perf Counter A31 Upper DWord
OAPERF_A32 - Aggregate_Perf_Counter_A32
OAPERF_A33 - Aggregate_Perf_Counter_A33
OAPERF_A34 - Aggregate_Perf_Counter_A34
OAPERF_A35 - Aggregate_Perf_Counter_A35
OAPERF_B0 - Boolean_Counter_B0
OAPERF_B1 - Boolean_Counter_B1
OAPERF_B2 - Boolean_Counter_B2
OAPERF_B3 - Boolean_Counter_B3
OAPERF_B4 - Boolean_Counter_B4
OAPERF_B5 - Boolean_Counter_B5
OAPERF_B6 - Boolean_Counter_B6
OAPERF_B7 - Boolean_Counter_B7
EU_PERF_CNT_CTL0 - Flexible EU Event Control 0
EU_PERF_CNT_CTL1 - Flexible EU Event Control 1
EU_PERF_CNT_CTL2 - Flexible EU Event Control 2
EU_PERF_CNT_CTL3 - Flexible EU Event Control 3
EU_PERF_CNT_CTL4 - Flexible EU Event Control 4
EU_PERF_CNT_CTL5 - Flexible EU Event Control 5
EU_PERF_CNT_CTL6 - Flexible EU Event Control 6

OA Interrupt Control Registers

The Interrupt Control Registers listed below all share the same bit definition. The bit definition is as follows:

Bit	Description
31:29	**Reserved. MBZ:** These bits may be assigned to interrupts on future products/steppings.
28	**Performance Monitoring Buffer Half-Full Interrupt:** For internal trigger (timer based) reporting, if the report buffer crosses the half full limit, this interrupt is generated.
27:0	**Reserved: MBZ (These bits must be never set by OA, these bit could be allocated to some other unit)**

- WDBoxOAInterrupt Vector
- IMR
- Bit Definition for Interrupt Control Registers

Performance Counter Report Formats

Counters layout for various values of select from the register:

Counters layout for various values of the "Counter Select" from the register:

Counter Select = 000

A-Cntr 10 (low dword)	A-Cntr 9 (low dword)	A-Cntr 8 (low dword)	A-Cntr 7 (low dword)	GPU_TICKS	CTX ID	TIME_STAMP	RPT_ID
A-Cntr 18 (low dword)	A-Cntr 17 (low dword)	A-Cntr 16 (low dword)	A-Cntr 15 (low dword)	A-Cntr 14 (low dword)	A-Cntr 13 (low dword)	A-Cntr 12 (low dword)	A-Cntr 11 (low dword)

Counter Select = 010

A-Cntr 10 (low dword)	A-Cntr 9 (low dword)	A-Cntr 8 (low dword)	A-Cntr 7 (low dword)	GPU_TICKS	CTX ID	TIME_STAMP	RPT_ID
A-Cntr 18 (low dword)	A-Cntr 17 (low dword)	A-Cntr 16 (low dword)	A-Cntr 15 (low dword)	A-Cntr 14 (low dword)	A-Cntr 13 (low dword)	A-Cntr 12 (low dword)	A-Cntr 11 (low dword)
B-Cntr 7	B-Cntr 6	B-Cntr 5	B-Cntr 4	B-Cntr 3	B-Cntr 2	B-Cntr 1	B-cntr 0
C-Cntr 7	C-Cntr 6	C-Cntr 5	C-Cntr 4	C-Cntr 3	C-Cntr 2	C-Cntr 1	C-Cntr 0

Counter Select = 111

C-Cntr 3	C-Cntr 2	C-Cntr 1	C-Cntr 0	GPU_TICKS	CTX ID	TIME_STAMP	RPT_ID
B-Cntr 7	B-Cntr 6	B-Cntr 5	B-Cntr 4	B-Cntr 3	B-Cntr 2	B-Cntr 1	B-Cntr 0

OACS Report Format (Counter Select = 0b101):

A-Cntr 3 (low dword)	A-Cntr 2 (low dword)	A-Cntr 1 (low dword)	A-Cntr 0 (low dword)	GPU_TICKS	CTX ID	TIME_STAMP	RPT_ID
A-Cntr 11 (low dword)	A-Cntr 10 (low dword)	A-Cntr 9 (low dword)	A-Cntr 8 (low dword)	A-Cntr 7 (low dword)	A-Cntr 6 (low dword)	A-Cntr 5 (low dword)	A-Cntr 4 (low dword)
A-Cntr 19 (low dword)	A-Cntr 18 (low dword)	A-Cntr 17 (low dword)	A-Cntr 16 (low dword)	A-Cntr 15 (low dword)	A-Cntr 14 (low dword)	A-Cntr 13 (low dword)	A-Cntr 12 (low dword)
A-Cntr 27 (low dword)	A-Cntr 26 (low dword)	A-Cntr 25 (low dword)	A-Cntr 24 (low dword)	A-Cntr 23 (low dword)	A-Cntr 22 (low dword)	A-Cntr 21 (low dword)	A-Cntr 20 (low dword)
A-Cntr 35 (low dword)	A-Cntr 34 (low dword)	A-Cntr 33 (low dword)	A-Cntr 32 (low dword)	A-Cntr 31 (low dword)	A-Cntr 30 (low dword)	A-Cntr 29 (low dword)	A-Cntr 28 (low dword)
High bytes of A31-A28	High bytes of A27-A24	High bytes of A23-A20	High bytes of A19-A16	High bytes of A15-A12	High bytes of A11-A8	High bytes of A7-A4	High bytes of A3-A0
B-Cntr 7	B-Cntr 6	B-Cntr 5	B-Cntr 4	B-Cntr 3	B-Cntr 2	B-Cntr 1	B-Cntr 0
C-Cntr 7	C-Cntr 6	C-Cntr 5	C-Cntr 4	C-Cntr 3	C-Cntr 2	C-Cntr 1	C-Cntr 0

Counter Select = 111

C-Cntr 3	C-Cntr 2	C-Cntr 1	C-Cntr 0	GPU_TICKS	CTX ID	TIME_STAMP	RPT_ID
B-Cntr 7	B-Cntr 6	B-Cntr 5	B-Cntr 4	B-Cntr 3	B-Cntr 2	B-Cntr 1	B-Cntr 0

Description of RPT_ID and other important fields of the layout:

Field	Description
GPU TICKS[31:0]	GPU_TICKS is simply a free-running count of render clocks elapsed used for normalizing other counters (e.g. EU active time), it is expected that the rate that this value advances will vary with frequency and freeze (but not lose its value) when all GT clocks are gated, GT is in RC6, and so on.
Context ID[31:0]	This field carries the Context ID of the active context in render/compute engine. [31:0]: Context ID in Execlist mode of scheduling.
TIME_STAMP[31:0]	This field provides an elapsed real-time value that can be used as a timestamp for GPU events over short periods of time. This field has the same format at TIMESTAMP register defined in

Field	Description
	Render Command Streamer.

SourceID[5:0]

Field	Description
GPU TICKS[31:0]	GPU_TICKS is simply a free-running count of render clocks elapsed used for normalizing other counters (e.g. EU active time), it is expected that the rate that this value advances will vary with frequency and freeze (but not lose its value) when all GT clocks are gated, GT is in RC6, and so on.
TIME_STAMP[31:0]	This field provides an elapsed real-time value that can be used as a timestamp for GPU events over short periods of time. This field has the same format at TIMESTAMP register defined in Render Command Streamer.
RPT_ID[31:0]	This field has several sub fields as defined below:

31:26	Reserved MBZ
25	**Render Context Valid:** When set indicates render context is valid and the ID is of the render context is set in “Context ID” field of report format.
24:19	**Report Reason[5:0]**: Report_reason[0]: When set indicates current report is due to “Timer Triggered”. Report_reason[1]: When set indicates current report is due to “Internal report trigger 1”. Report_reason[2]: When set indicates current report is due to “Internal report trigger 2”. Report_reason[3]: When set indicates current report is due to “Render context switch”. Report_reason[4]: When set indicates current report is due to “GO transition from ‘1’ to ‘0’ ”. Report_reason[5]: Reserved
18	**Start Trigger Event:**This bit is multiplexed from “Start Trigger Event-1” or “Start Trigger Event-2” based on the “Internal Report Trigger-1” or “Internal Report Trigger-2” asserted in the Report Reason respectively. “Internal Report Trigger-1” is given priority over “Internal Report Trigger-2”. By default Start Trigger Event-1 is outputted.
17	**Threshold Enable:** This bit is multiplexed from “Report Trigger Threshold Enable-1” or “Report Trigger Threshold Enable-2” based on the “Internal Report Trigger-1” or “Internal Report Trigger-2” asserted in the Report Reason respectively. “Internal Report Trigger-1” is given priority over “Internal Report Trigger-2”. By default “Report Trigger Threshold Enable-1” is outputted.
16	**Timer Enabled**
15:0	Reserved

RPT_ID[31:0]	Subfields of RPT_ID detailed below:

31:25	**squashed_slice_clock_frequency[6:0]**:

Field	Description	
		Ratio encoding in this field can be decoded using the ratio encoding table that is part of the definition of the RP_FREQ_NORMAL register.
	24:19	**Report Reason[5:0]**: Report_reason[0]: When set indicates current report is due to "Timer Triggered". Report_reason[1]: When set indicates current report is due to "Internal report trigger 1". Report_reason[2]: When set indicates current report is due to "Internal report trigger 2". Report_reason[3]: When set indicates current report is due to "Render context switch". Report_reason[4]: When set indicates current report is due to "GO transition from '1' to '0' ". Report_reason[5]: When set indicates the current report is due to Clock Ratio change between squashed Slice Clock frequency to squashed Unslice clock frequency.
	18	**Start Trigger Event:**This bit is multiplexed from "Start Trigger Event-1" or "Start Trigger Event-2" based on the "Internal Report Trigger-1" or "Internal Report Trigger-2" asserted in the Report Reason respectively. "Internal Report Trigger-1" is given priority over "Internal Report Trigger-2". By default Start Trigger Event-1 is outputted.
	17	**Threshold Enable:** This bit is multiplexed from "Report Trigger Threshold Enable-1" or "Report Trigger Threshold Enable-2" based on the "Internal Report Trigger-1" or "Internal Report Trigger-2" asserted in the Report Reason respectively. "Internal Report Trigger-1" is given priority over "Internal Report Trigger-2". By default "Report Trigger Threshold Enable-1" is outputted.
	16	**Render Context Valid:** When set indicates render context is valid and the ID is of the render context is set in "Context ID" field of report format.
	15:0	**Reserved**

Performance Counter Reporting

When either the MI_REPORT_PERF_COUNT command is received or the internal report trigger logic fires, a snapshot of the performance counter values is written to memory. The format used by HW for such reports is selected using the Counter Select field within the **OACONTROL** register. The organization and number of report formats vary per project and are detailed in Performance Counter Report Formats.

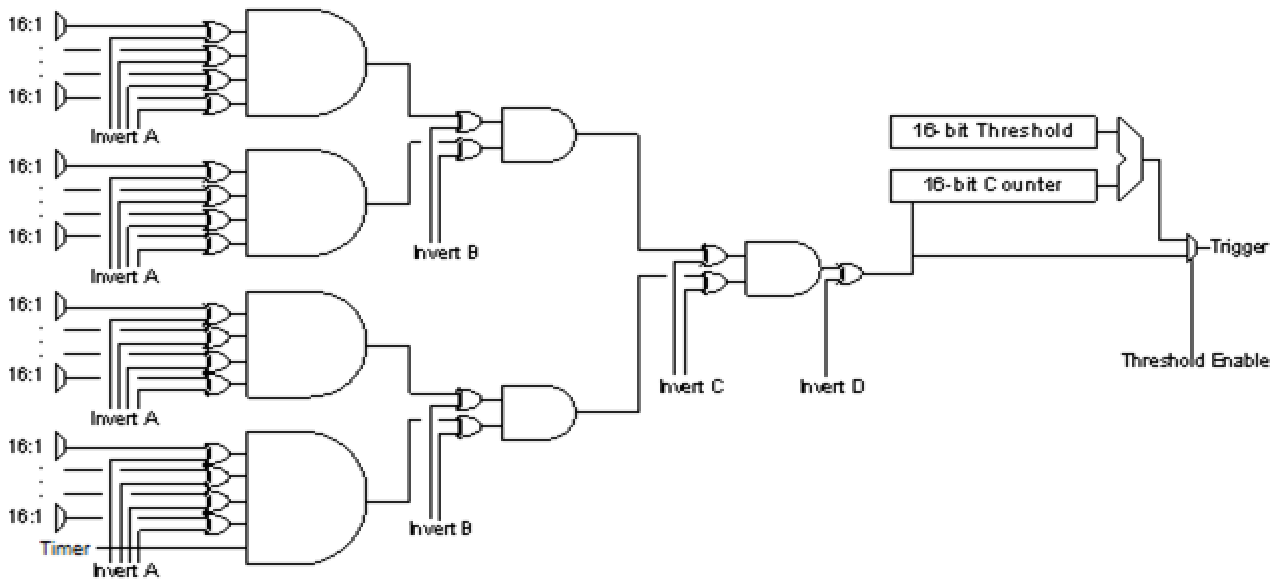
Details of Start Trigger Behavior

- All counters not explicitly defined as free-running will advance after the start trigger conditions are met.
- Counting will continue after the start trigger has fired until OA is disabled or device is reset.
- Multiple start triggering blocks (where implemented) are OR'd together in order to allow specification of multiple trigger conditions.
- Bit 18 in the report format reflects whether the start trigger has fired or not.

While architectural intent was that Start Trigger logic would control all qualified counter types (A/B/C), there is a long-standing implementation bug whereby start trigger logic only affects B/C counters.

Configuration of Trigger Logic

OA contains logic to control when performance counter values are reported to memory. This functionality is controlled using the OA report trigger and OA start trigger registers. More detailed register descriptions are included in the Hardware Programming interface. The block diagram below illustrates the logic these registers control.



Note that counters which are 40 bits wide are split in the report format into low DWORD and high byte chunks for simplicity of HW implementation as well as SW-friendly alignment of report data. The performance counter read logically done before writing out report data for these 40-bit counters is guaranteed to be an atomic operation, the counter data is simply swizzled as it is being packed into the report.

Context Switch Triggered Reports

A context load/switch on RCS will cause a performance counter snapshot to be written to memory at the next location in the OA circular report buffer using the perf counter format selected in OACONTROL (**[Register] Observation Architecture Control**). This functionality can be leveraged when preemption is enabled to re-construct the contribution of a specific context to a performance counter delta, requires SW to consider both the delta reported by MI_REPORT_PERF and the reports that may have been issued to OABUFFER by intervening contexts.

Frequency Change Triggered Reports

A GFX frequency change will cause a performance counter snapshot to be written to memory at the next location in the OA circular report buffer using the perf counter format selected in OACONTROL (**[Register] Observation Architecture Control**). Please note that a change back to the same frequency can occur and that such changes will still cause a performance counter report to occur.

Aggregating Counters

The table below described the desired high-level functionality from each of the aggregating counters.

Note that there is no counter of 2x2s sent to pixel shader, this is based on the assumption that the pixel shader invocation pipeline statistics counter increments for partially lit 2x2s as well and hence does not require a duplicate performance counter.

Please also note that some of the information provided by A-counters is useful for GFX/system load-balancing and is hence made available via free-running counters which do not require initial setup and count irrespective of OA enable/disable or freeze.

Counter #	Event	Description
A0	GPU Busy	GPU is not idle (includes all GPU engines). Link to detailed register definition: **[Register] Aggregate Perf Counter A0**
A1	# of Vertex Shader Threads Dispatched	Count of VS threads dispatched to EUs Link to detailed register definition: **[Register] Aggregate Perf Counter A1**
A2	# of Hull Shader Threads Dispatched	Count of HS threads dispatched to EUs Link to detailed register definition: **[Register] Aggregate Perf Counter A2**
A3	# of Domain Shader Threads Dispatched	Count of DS threads dispatched to EUs Link to detailed register definition: **[Register] Aggregate Perf Counter A3**
A4	# of GPGPU Threads Dispatched	Count of GPGPU threads dispatched to EUs Available on both qualified and free-running counters Link to detailed register definition: **[Register] Aggregate Perf Counter A4**
A5	# of Geometry Shader Threads Dispatched	Count of GS threads dispatched to EUs Link to detailed register definition: **[Register] Aggregate Perf Counter A5**
A6	# of Pixel Shader Threads Dispatched	Count of PS threads dispatched to EUs Available on both qualified and free-running counters Link to detailed register definition:

Counter #	Event	Description
		[Register] Aggregate Perf Counter A6
A7	Aggregating EU counter 0	User-defined (details in Flexible EU Event Counters section) Link to detailed register definition: **[Register] Aggregate Perf Counter A7**
A8	Aggregating EU counter 1	User-defined (details in Flexible EU Event Counters section) Link to detailed register definition: **[Register] Aggregate Perf Counter A8**
A9	Aggregating EU counter 2	User-defined (details in Flexible EU Event Counters section) Link to detailed register definition: **[Register] Aggregate Perf Counter A9**
A10	Aggregating EU counter 3	User-defined (details in Flexible EU Event Counters section) Link to detailed register definition: **[Register] Aggregate Perf Counter A10**
A11	Aggregating EU counter 4	User-defined (details in Flexible EU Event Counters section) Link to detailed register definition: **[Register] Aggregate Perf Counter A11**
A12	Aggregating EU counter 5	User-defined (details in Flexible EU Event Counters section) Link to detailed register definition: **[Register] Aggregate Perf Counter A12**
A13	Aggregating EU counter 6	User-defined (details in Flexible EU Event Counters section) Link to detailed register definition: **[Register] Aggregate Perf Counter A13**
A14	Aggregating EU counter 7	User-defined (details in Flexible EU Event Counters section) Link to detailed register definition: **[Register] Aggregate Perf Counter A14**
A15	Aggregating EU counter 8	User-defined (details in Flexible EU Event Counters section) Link to detailed register definition: **[Register] Aggregate Perf Counter A15**
A16	Aggregating EU counter 9	User-defined (details in Flexible EU Event Counters section) Link to detailed register definition:

Counter #	Event	Description
		[Register] Aggregate Perf Counter A16
A17	Aggregating EU counter 10	User-defined (details in Flexible EU Event Counters section) Link to detailed register definition: **[Register] Aggregate Perf Counter A17**
A18	Aggregating EU counter 11	User-defined (details in Flexible EU Event Counters section) Link to detailed register definition: **[Register] Aggregate Perf Counter A18**
A19	Aggregating EU counter 12	Available on both qualified and free-running counters User-defined (details in Flexible EU Event Counters section) Link to detailed register definition: **[Register] Aggregate Perf Counter A19**
A20	Aggregating EU counter 13	Available on both qualified and free-running counters User-defined (details in Flexible EU Event Counters section) Link to detailed register definition: **[Register] Aggregate Perf Counter A20**
A21	2x2s Rasterized	Count of the number of samples of 2x2 pixel blocks generated from the input geometry before any pixel-level tests have been applied. (Please note that 2x2s may be in terms of pixels or in terms of samples depending on project but are consistent between A21-A27.) Link to detailed register definition: **[Register] Aggregate Perf Counter A21**
A22	2x2s Failing Fast pre-PS Tests	Count of the number of samples failing fast "early" (i.e. before pixel shader execution) tests (counted at 2x2 granularity). (Please note that 2x2s may be in terms of pixels or in terms of samples depending on project but are consistent between A21-A27.) Link to detailed register definition: **[Register] Aggregate Perf Counter A22**
A23	2x2s Failing Slow pre-PS Tests	Count of the number of samples of failing slow "early" (i.e. before pixel shader execution) tests (counted at 2x2 granularity). (Please note that 2x2s may be in terms of pixels or in terms of samples depending on project but are consistent between A21-A27.) Link to detailed register definition: **[Register] Aggregate Perf Counter A23**

Counter #	Event	Description
A24	2x2s Killed in PS	Number of samples entirely killed in the pixel shader as a result of explicit instructions in the kernel (counted in 2x2 granularity). (Please note that 2x2s may be in terms of pixels or in terms of samples depending on project but are consistent between A21-A27.) Link to detailed register definition: **[Register] Aggregate Perf Counter A24**
A25	2x2s Failing post-PS Tests	Number of samples that entirely fail "late" tests (i.e. tests that can only be performed after pixel shader execution). Counted at 2x2 granularity. (Please note that 2x2s may be in terms of pixels or in terms of samples depending on project but are consistent between A21-A27.) Link to detailed register definition: **[Register] Aggregate Perf Counter A25**
A26	2x2s Written To Render Target	Number of samples that are written to render target.(counted at 2x2 granularity). MRT case will report multiple writes per 2x2 processed by the pixel shader. (Please note that 2x2s may be in terms of pixels or in terms of samples depending on project but are consistent between A21-A27.) Please note that this counter will not advance if a render target update does not occur and that pixel masking operations performed by the fixed function HW or shader may not be reflected in counters A22-A25 which only track their specific defined operations. This can lead to an apparent discrepancy between A21 vs. A22-A25 vs. A26/A27. Link to detailed register definition: **[Register] Aggregate Perf Counter A26**
A27	Blended 2x2s Written to Render Target	Number of samples of blendable that are written to render target.(counted at 2x2 granularity). MRT case will report multiple writes per 2x2 processed by the pixel shader. (Please note that 2x2s may be in terms of pixels or in terms of samples depending on project but are consistent between A21-A27.) Please note that this counter will not advance if a render target update does not occur and that pixel masking operations performed by the fixed function HW or shader may not be reflected in counters A22-A25 which only track their specific defined operations. This can lead to an apparent discrepancy between A21 vs. A22-A25 vs. A26/A27. Link to detailed register definition: **[Register] Aggregate Perf Counter A27**
A28	2x2s Requested from Sampler	Aggregated total 2x2 texel blocks requested from all EUs to all instances of sampler logic.

Counter #	Event	Description
		Link to detailed register definition: **[Register] Aggregate Perf Counter A28**
A29	Sampler L1 Misses	Aggregated misses from all sampler L1 caches. Please note that the number of L1 accesses varies with requested filtering mode and in other implementation specific ways. Hence it is not possible in general to draw a direct relationship between A28 and A29. However, a high number of sampler L1 misses relative to texel 2x2s requested frequently degrades sampler performance. Link to detailed register definition: **[Register] Aggregate Perf Counter A29**
A30	SLM Reads	Total read requests from an EU to SLM (including reads generated by atomic operations). Link to detailed register definition: **[Register] Aggregate Perf Counter A30**
A31	SLM Writes	Total write requests from an EU to SLM (including writes generated by atomic operations). Link to detailed register definition: **[Register] Aggregate_Perf_Counter_A31**
A32	Other Shader Memory Accesses	Aggregated total requests from all EUs to memory surfaces other than render target or texture surfaces (e.g. shader constants). Link to detailed register definition: **[Register] Aggregate_Perf_Counter_A32**
A33	Other Shader Memory Accesses That Miss First-Level Cache	Aggregated total requests from all EUs to memory surfaces other than render target or texture surfaces (e.g. shader constants) that miss first-level cache. Link to detailed register definition: **[Register] Aggregate_Perf_Counter_A33**
A34	Atomic Accesses	Aggregated total atomic accesses from all EUs. This counter increments on atomic accesses to both SLM and URB. Link to detailed register definition: **[Register] Aggregate_Perf_Counter_A34** **Workaround** SLM atomics are not included by this OA event (only global memory atomics are counted), a workaround using B/C counters is possible.
A35	Barrier Messages	Aggregated total kernel barrier messages from all Eus (one per thread in barrier).

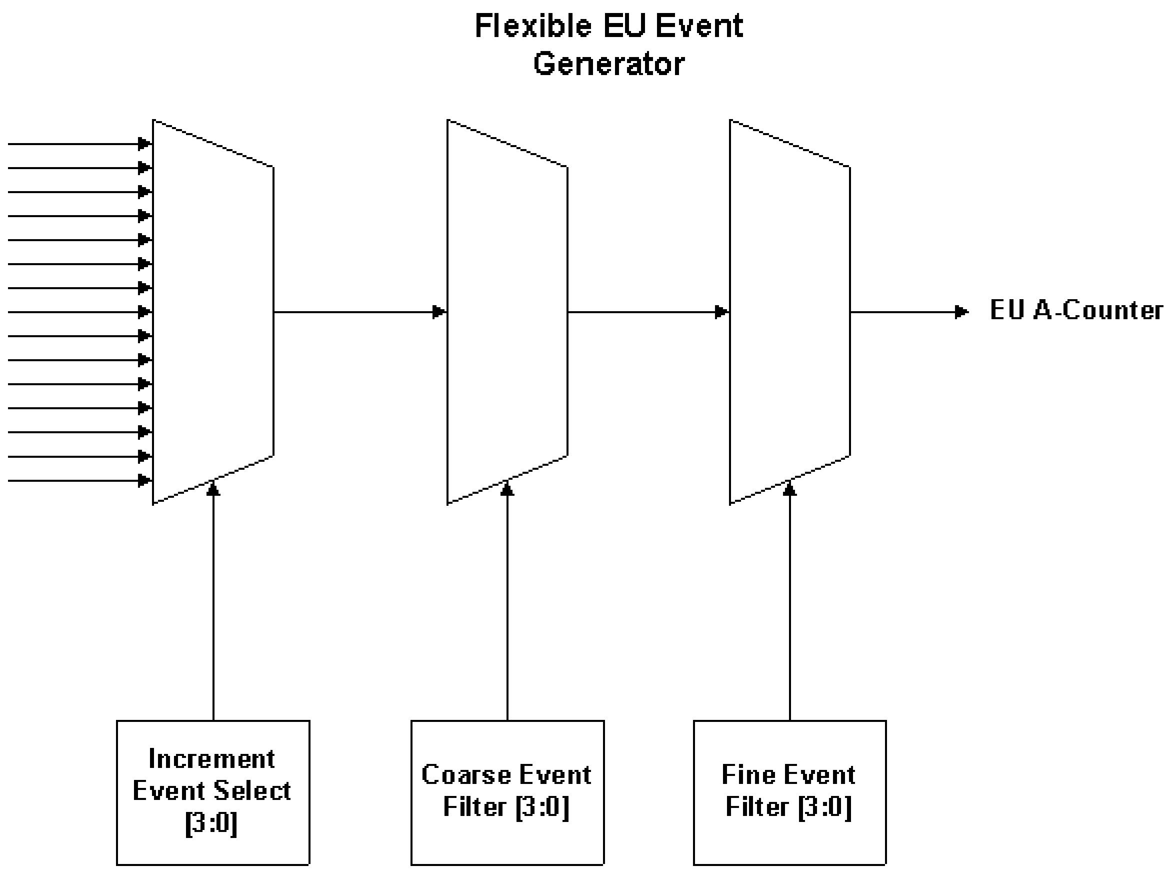
Counter #	Event	Description
		Link to detailed register definition: **[Register] Aggregate_Perf_Counter_A35**

Flexible EU Event Counters

Since EU performance events are most interesting in many cases when aggregated across all EUs and many interesting EU performance events are limited to certain APIs (e.g. hull shader kernel stats only applicable when running a DX11+ workload), adds some additional flexibility to the aggregated counters coming from the EU array.

The following block diagram shows the high-level flow that generates each flexible EU event.

Note that no support is provided for differences between flexible EU event programming between EUs because the resulting output from each EU is eventually merged into a single OA counter anyway.


Flexible EU Event Generator
EU A-Counter
Increment Event Select [3:0]
Coarse Event Filter [3:0]
Fine Event Filter [3:0]

Supported Increment Events

Increment Event	Encoding	Notes
EU FPU0 Pipeline Active	0b0000	Signal that is high on every EU clock where the EU FPU0 pipeline is actively executing an ISA instruction.
EU FPU1 Pipeline Active	0b0001	Signal that is high on every EU clock where the EU FPU1 pipeline is actively executing an ISA instruction.
EU SEND Pipeline Active	0b0010	Signal that is high on every EU clock where the EU send pipeline is actively executing an ISA instruction. Only fine event filters 0b0000,0b0101, 0b0111, 0b1000, 0b1001, and 0b1010 are supported with this increment event.
EU FPU0 & FPU1 Pipelines Concurrently Active	0b0011	Signal that is high on every EU clock where the EU FPU0 and FPU1 pipelines are both actively executing an ISA instruction. Only coarse event filters 0b0000, 0b0111, and 0b1000 are supported with this increment event. Only fine event filters 0b0000, 0b0111, 0b1000, 0b1001, and 0b1010 are supported with this increment event.
Some EU Pipeline Active	0b0100	Signal that is high on every EU clock where at least one EU pipeline is actively executing an ISA instruction. Only coarse event filters 0b0000, 0b0111, and 0b1000 are supported with this increment event. Only fine event filters 0b0000,0b0101, 0b0111, 0b1000, 0b1001, and 0b1010 are supported with this increment event.
At Least 1 Thread Loaded But No EU Pipeline Active	0b0101	Signal that is high on every EU clock where at least one thread is loaded but no EU pipeline is actively executing an ISA instruction. Only coarse event filters 0b0000, 0b0111, and 0b1000 are supported with this increment event. Only fine event filters 0b0000, 0b0111, 0b1000, 0b1001, and 0b1010 are supported with this increment event.
Threads loaded integrator == max threads for current HW SKU	0b1000	Implies an accumulator which increases every EU clock by the number of loaded threads, signal pulses high for one clock when the accumulator exceeds a multiple of the number of thread slots (e.g. for a 8-thread EU, signal pulses high every clock where the increment causes a 3-bit accumulator to overflow). Only coarse event filters 0b0000, 0b0111, and 0b1000 are supported with this increment event. Only fine event filters 0b0000, 0b0111, 0b1000, 0b1001, and 0b1010 are supported with this increment event.
No event	0b1111	Expected HW default, allows logic to be power-optimized.

Supported Coarse Event Filters

Coarse Event Filter	Encoding	Notes
No mask	0b0000	Never masks increment event.
VS Thread Filter	0b0001	For increment events 0b0000/0b0001/0b0010, masks increment events unless the FFID which dispatched the currently executing thread equals FFID of VS.
HS Thread Filter	0b0010	For increment events 0b0000/0b0001/0b0010, masks increment event unless the FFID which dispatched the currently executing thread equals FFID of HS.

Coarse Event Filter	Encoding	Notes
DS Thread Filter	0b0011	For increment events 0b0000/0b0001/0b0010, masks increment event unless the FFID which dispatched the currently executing thread equals FFID of DS.
GS Thread Filter	0b0100	For increment events 0b0000/0b0001/0b0010, masks increment event unless the FFID which dispatched the currently executing thread equals FFID of GS.
PS Thread Filter	0b0101	For increment events 0b0000/0b0001/0b0010, masks increment event unless the FFID which dispatched the currently executing thread equals FFID of PS.
TS Thread Filter	0b0110	For increment events 0b0000/0b0001/0b0010, masks increment event unless the FFID which dispatched the currently executing thread equals FFID of TS.
Row = 0	0b0111	Masks increment event unless the row ID for this EU is 0 (control register is in TDL so only have to check within quarter-slice).
Row = 1	0b1000	Masks increment event unless the row ID for this EU is 1 (control register is in TDL so only have to check within quarter-slice).

Fine Event Filters

Fine Event Filter	Encoding	Notes
None	0b0000	Never mask increment event.
Cycles where hybrid instructions are being executed	0b0001	Masks increment event unless the instruction(s) being executed on the pipeline(s) selected by the increment event are hybrid instructions.
Cycles where ternary instructions are being executed	0b0010	Masks increment event unless the instruction(s) being executed on the pipeline(s) selected by the increment event are ternary instructions.
Cycles where binary instructions are being executed	0b0011	Masks increment event unless the instruction(s) being executed on the pipeline(s) selected by the increment event are binary instructions.
Cycles where mov instructions are being executed	0b0100	Masks increment event unless the instruction(s) being executed on the pipeline(s) selected by the increment event are mov instructions.
Cycles where sends start being executed	0b0101	Masks increment event unless the instruction(s) being executed on the pipeline(s) selected by the increment event are send start of dispatch. Note that if this fine event filter is used in combination with increment events not related to the EU send pipeline (e.g. FPU0 active), the associated flexible event counter will increment in an implementation-specific manner.
EU# = 0b00	0b0111	Masks increment event unless the EU number for this EU is 0b00.
EU# = 0b01	0b1000	Masks increment event unless the EU number for this EU is 0b01.
EU# = 0b10	0b1001	Masks increment event unless the EU number for this EU is 0b10.
EU# = 0b11	0b1010	Masks increment event unless the EU number for this EU is 0b11.

Flexible EU Event Config Registers

EU_PERF_CNT_CTL0 - Flexible EU Event Control 0

EU_PERF_CNT_CTL1 - Flexible EU Event Control 1

EU_PERF_CNT_CTL2 - Flexible EU Event Control 2

EU_PERF_CNT_CTL3 - Flexible EU Event Control 3

EU_PERF_CNT_CTL4 - Flexible EU Event Control 4

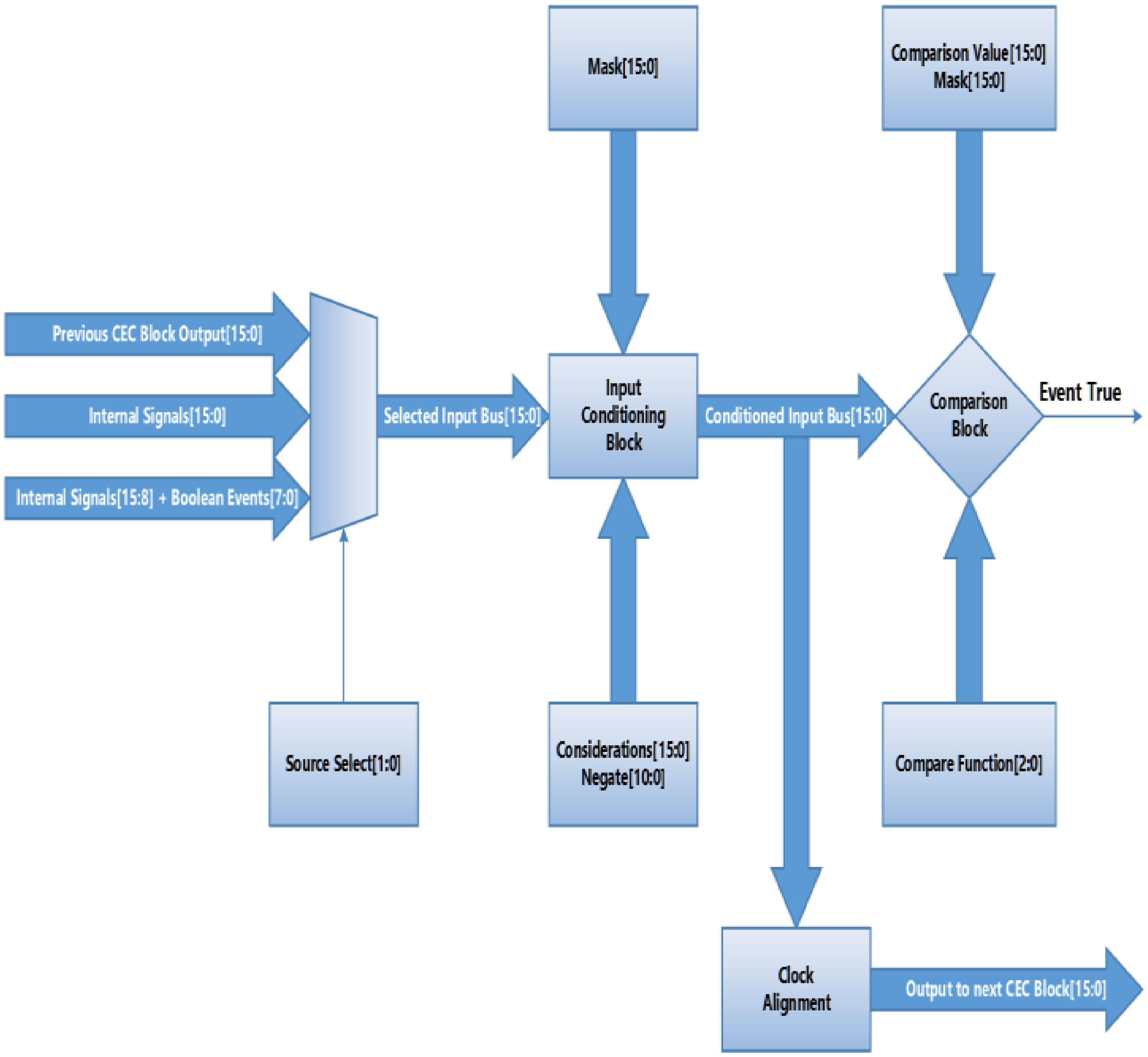
EU_PERF_CNT_CTL5 - Flexible EU Event Control 5

EU_PERF_CNT_CTL5 - Flexible EU Event Control 6

Custom Event Counters

Also known as B-counters, the events counted in these counters are defined from Boolean combinations of input signals using the custom event creation logic built into OA.

The following diagram(s) illustrate(s) the structure used to create a custom event. Every B-counter has such a block.



MI_REPORT_PERF_COUNT