



# **Intel® Open Source HD Graphics, Intel Iris™ Graphics, and Intel Iris™ Pro Graphics**

## **Programmer's Reference Manual**

For the 2015 - 2016 Intel Core™ Processors, Celeron™ Processors, and Pentium™ Processors based on the "Skylake" Platform

Volume 14: Observability

May 2016, Revision 1.0

## Creative Commons License

**You are free to Share** - to copy, distribute, display, and perform the work under the following conditions:

- **Attribution.** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **No Derivative Works.** You may not alter, transform, or build upon this work.

## Notices and Disclaimers

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

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

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Implementations of the I2C bus/protocol may require licenses from various entities, including Philips Electronics N.V. and North American Philips Corporation.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

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

**Copyright © 2016, Intel Corporation. All rights reserved.**

## Table of Contents

<b>Observability Overview</b> .....	<b>1</b>
<b>Trace</b> .....	<b>1</b>
Performance Visibility.....	1
Motivation for Hardware-Assisted Performance Visibility .....	1
Performance Event Counting.....	1
HW Support.....	2
Performance Counter Registers .....	2
OA Interrupt Control Registers .....	5
Performance Counter Report Formats .....	6
Performance Counter Reporting .....	10
Aggregating Counters .....	11
Flexible EU Event Counters .....	13
MI_REPORT_PERF_COUNT.....	16



## 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.

## Trace

This section contains the following contents:

Feature
<ul style="list-style-type: none"> <li>Performance Visibility</li> </ul>

## 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.

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 pin point 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.

Gen7 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.

Gen7.5 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.

Gen8 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.

## HW Support

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

### Performance Counter Registers

Register
<b>OACONTROL - Observation Architecture Control</b>
<b>OACTXCONTROL - Observation Architecture Control per Context</b>
<b>OACTXID - Observation Architecture Control Context ID</b>
<b>OA_IMR - OA Interrupt Mask Register</b>
<b>OASTATUS - Observation Architecture Status Register</b>
<b>OAHEADPTR - Observation Architecture Head Pointer</b>
<b>OATAILPTR - Observation Architecture Tail Pointer</b>
<b>OABUFFER - Observation Architecture Buffer</b>
<b>OASTARTTRIG_COUNTER - Observation Architecture Start Trigger Counter</b>
<b>OARPTTRIG_COUNTER - Observation Architecture Report Trigger Counter</b>
<b>OAREPORTTRIG2 - Observation Architecture Report Trigger 2</b>
<b>OAREPORTTRIG6 - Observation Architecture Report Trigger 6</b>
<b>CEC0-0 - Customizable Event Creation 0-0</b>
<b>CEC1-0 - Customizable Event Creation 1-0</b>

Register
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 2
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
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 - 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

Register
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
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



Register
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
GPU_TICKS - GPU_Ticks_Counter

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

- **GT Interrupt 3 Definition**
- 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 = 101

#### Performance Counter Report Format 101b

#### 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 engine. [31:0]: Context ID in Execlist mode of scheduling. [31:12]: Context ID in Ring Buffer mode of scheduling, [11:0] must be ignored.

Field	Description														
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 as the TIME_STAMP register defined in Vol1C.4 Render Command Streamer BSpec.														
RPT_ID[31:0]	<p>This field has several sub fields as defined below:</p> <table border="1" data-bbox="375 449 1498 1417"> <tr> <td data-bbox="375 449 456 491">31:26</td> <td data-bbox="456 449 1498 491">Reserved MBZ</td> </tr> <tr> <td data-bbox="375 491 456 575">25</td> <td data-bbox="456 491 1498 575"><b>Render Context Valid:</b> When set indicates render context is valid and the ID is of the render context is set in "Context ID" field of report format.</td> </tr> <tr> <td data-bbox="375 575 456 995">24:19</td> <td data-bbox="456 575 1498 995"> <p><b>Report Reason[5:0]:</b></p> <p>Report_reason[0]: When set indicates current report is due to "Timer Triggered".</p> <p>Report_reason[1]: When set indicates current report is due to "Internal report trigger 1".</p> <p>Report_reason[2]: When set indicates current report is due to "Internal report trigger 2".</p> <p>Report_reason[3]: When set indicates current report is due to "Render context switch".</p> <p>Report_reason[4]: When set indicates current report is due to "GO transition from '1' to '0'".</p> <p>Report_reason[5]: Reserved</p> </td> </tr> <tr> <td data-bbox="375 995 456 1142">18</td> <td data-bbox="456 995 1498 1142"><b>Start Trigger Event:</b> 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.</td> </tr> <tr> <td data-bbox="375 1142 456 1331">17</td> <td data-bbox="456 1142 1498 1331"><b>Threshold Enable:</b> 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.</td> </tr> <tr> <td data-bbox="375 1331 456 1373">16</td> <td data-bbox="456 1331 1498 1373"><b>Timer Enabled</b></td> </tr> <tr> <td data-bbox="375 1373 456 1417">15:0</td> <td data-bbox="456 1373 1498 1417"></td> </tr> </table>	31:26	Reserved MBZ	25	<b>Render Context Valid:</b> 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	<p><b>Report Reason[5:0]:</b></p> <p>Report_reason[0]: When set indicates current report is due to "Timer Triggered".</p> <p>Report_reason[1]: When set indicates current report is due to "Internal report trigger 1".</p> <p>Report_reason[2]: When set indicates current report is due to "Internal report trigger 2".</p> <p>Report_reason[3]: When set indicates current report is due to "Render context switch".</p> <p>Report_reason[4]: When set indicates current report is due to "GO transition from '1' to '0'".</p> <p>Report_reason[5]: Reserved</p>	18	<b>Start Trigger Event:</b> 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	<b>Threshold Enable:</b> 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	<b>Timer Enabled</b>	15:0	
31:26	Reserved MBZ														
25	<b>Render Context Valid:</b> 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	<p><b>Report Reason[5:0]:</b></p> <p>Report_reason[0]: When set indicates current report is due to "Timer Triggered".</p> <p>Report_reason[1]: When set indicates current report is due to "Internal report trigger 1".</p> <p>Report_reason[2]: When set indicates current report is due to "Internal report trigger 2".</p> <p>Report_reason[3]: When set indicates current report is due to "Render context switch".</p> <p>Report_reason[4]: When set indicates current report is due to "GO transition from '1' to '0'".</p> <p>Report_reason[5]: Reserved</p>														
18	<b>Start Trigger Event:</b> 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	<b>Threshold Enable:</b> 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	<b>Timer Enabled</b>														
15:0															
RPT_ID[31:0]	<p>Subfields of RPT_ID detailed below:</p> <table border="1" data-bbox="375 1499 1498 1581"> <tr> <td data-bbox="375 1499 456 1581">31:25</td> <td data-bbox="456 1499 1498 1581"><b>squashed_slice_clock_frequency[6:0]:</b></td> </tr> </table>	31:25	<b>squashed_slice_clock_frequency[6:0]:</b>												
31:25	<b>squashed_slice_clock_frequency[6:0]:</b>														

Field	Description
24:19	<p><b>Report Reason[5:0]:</b>            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.</p>
18	<p><b>Start Trigger Event:</b> 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.</p>
17	<p><b>Threshold Enable:</b> 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.</p>
16	<p><b>Render Context Valid:</b> When set indicates render context is valid and the ID is of the render context is set in "Context ID" field of report format.</p>
15:0	<p><b>Additional Report Flags:</b>            [10:9]: squashed_slice_clock_frequency [8:7]            [8:0]: squashed_unslice_clock_frequency [8:0]</p>

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 Vol1C.4 Render Command Streamer BSpec.				
RPT_ID[31:0]	<p>This field has several sub fields as defined below:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50px;">31:26</td> <td>Reserved MBZ</td> </tr> <tr> <td>25</td> <td><b>Render Context Valid:</b> When set indicates render context is valid and the ID is of the render context is set in "Context ID" field of report format.</td> </tr> </table>	31:26	Reserved MBZ	25	<b>Render Context Valid:</b> When set indicates render context is valid and the ID is of the render context is set in "Context ID" field of report format.
31:26	Reserved MBZ				
25	<b>Render Context Valid:</b> When set indicates render context is valid and the ID is of the render context is set in "Context ID" field of report format.				

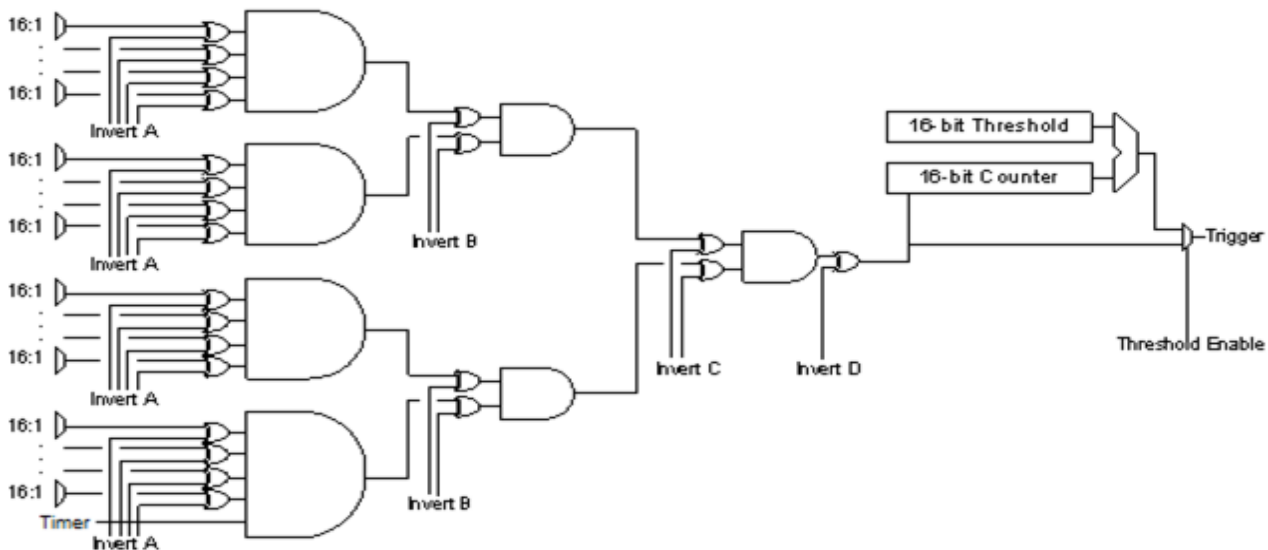
Field	Description
	24:19 <b>Report Reason[5:0]:</b> Report_reason[0]: When set indicates current report is due to "Timer Triggered". Report_reason[1]: When set indicates current report is due to "Internal report trigger 1". Report_reason[2]: When set indicates current report is due to "Internal report 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 <b>Start Trigger Event:</b> 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 <b>Threshold Enable:</b> 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 <b>Timer Enabled</b>
	15:0
RPT_ID[31:0]	Subfields of RPT_ID detailed below:
	31:25 <b>squashed_slice_clock_frequency[6:0]:</b>
	24:19 <b>Report Reason[5:0]:</b> 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 <b>Start Trigger Event:</b> 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.

Field	Description
17	<b>Threshold Enable:</b> 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	<b>Render Context Valid:</b> 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	<b>Additional Report Flags:</b> [10:9]: squashed_slice_clock_frequency [8:7] [8:0]: squashed_unslice_clock_frequency [8:0]

### 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 the following section. In the following layouts, the RPT\_ID is always stored in the lowest addressed DWORD.

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.

## 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.

Counter #	Event	Description
A0	Render Engine Busy	Render engine is not idle. GPU Busy aggregate counter doesn't increment under the following conditions: <ol style="list-style-type: none"> <li>Context Switch in Progress.</li> <li>GPU stalled on executing MI_WAIT_FOR_EVENT.</li> <li>GPU stalled on execution MI_SEMAPHORE_MBOX.</li> <li>RCS idle but other parts of GPU active (e.g. only media engines active)</li> </ol>
A1	# of Vertex Shader Threads Dispatched	Count of VS threads dispatched to EUs
A2	# of Hull Shader Threads Dispatched	Count of HS threads dispatched to EUs
A3	# of Domain Shader Threads Dispatched	Count of DS threads dispatched to EUs
A4	# of GPGPU Threads Dispatched	Count of GPGPU threads dispatched to EUs
A5	# of Geometry Shader Threads Dispatched	Count of GS threads dispatched to EUs
A6	# of Pixel Shader Threads Dispatched	Count of PS threads dispatched to EUs
A7	Aggregating EU counter 0	User-defined (details in <a href="#">Flexible EU Event Counters</a> section)
A8	Aggregating EU counter 1	User-defined (details in <a href="#">Flexible EU Event Counters</a> section)
A9	Aggregating EU counter 2	User-defined (details in <a href="#">Flexible EU Event Counters</a> section)
A10	Aggregating EU counter 3	User-defined (details in <a href="#">Flexible EU Event Counters</a> section)
A11	Aggregating EU counter 4	User-defined (details in <a href="#">Flexible EU Event Counters</a> section)
A12	Aggregating EU counter 5	User-defined (details in <a href="#">Flexible EU Event Counters</a> section)
A13	Aggregating EU counter 6	User-defined (details in <a href="#">Flexible EU Event Counters</a> section)

Counter #	Event	Description
A14	Aggregating EU counter 7	User-defined (details in <a href="#">Flexible EU Event Counters</a> section)
A15	Aggregating EU counter 8	User-defined (details in <a href="#">Flexible EU Event Counters</a> section)
A16	Aggregating EU counter 9	User-defined (details in <a href="#">Flexible EU Event Counters</a> section)
A17	Aggregating EU counter 10	User-defined (details in <a href="#">Flexible EU Event Counters</a> section)
A18	Aggregating EU counter 11	User-defined (details in <a href="#">Flexible EU Event Counters</a> section)
A19	Aggregating EU counter 12	User-defined (details in <a href="#">Flexible EU Event Counters</a> section)
A20	Aggregating EU counter 13	User-defined (details in <a href="#">Flexible EU Event Counters</a> section)
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.)
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.)
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.)
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.)  Counter may be inaccurate when pixel shader outputs output mask (e.g. DX11 oMask declaration)
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.)
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.)
A26	2x2s Written To	Number of samples that are written to render target.(counted at 2x2 granularity).



Counter #	Event	Description
	Render Target	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.)
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.)
A28	2x2s Requested from Sampler	Aggregated total 2x2 texel blocks requested from all EUs to all instances of sampler logic.
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.
A30	SLM Reads	Total read requests from an EU to SLM (including reads generated by atomic operations).
A31	SLM Writes	Total write requests from an EU to SLM (including writes generated by atomic operations).
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).
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.
A34	Atomic Accesses	Aggregated total atomic accesses from all EUs. This counter increments on atomic accesses to both SLM and URB.
A35	Barrier Messages	Aggregated total kernel barrier messages from all Eus (one per thread in barrier).

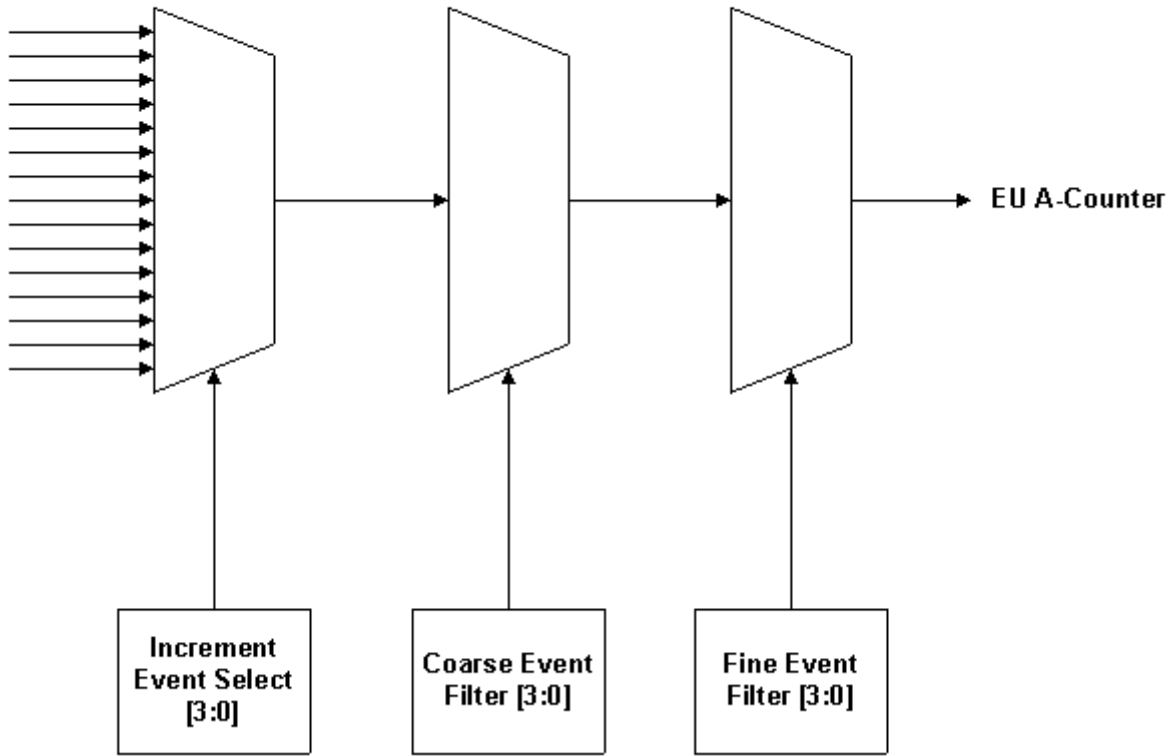
## Flexible EU Event Counters

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



### 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 a Gen ISA instruction.
EU FPU1 Pipeline Active	0b0001	Signal that is high on every EU clock where the EU FPU1 pipeline is actively executing a Gen ISA instruction.
		Event doesn't count the second pass of multi-pass extended math instructions. Hence, it counts only 1/2 of the desired utilization while the EU is executing these instructions.
EU SEND Pipeline Active	0b0010	Signal that is high on every EU clock where the EU send pipeline is actively executing a Gen ISA instruction. Only fine event filters 0b0000, 0b0101, 0b0110, 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 a Gen 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 a Gen ISA instruction. Only coarse event filters 0b0000, 0b0111, and 0b1000 are supported with this increment event. Only fine event filters 0b0000, 0b0101, 0b0110, 0b0111, 0b1000, 0b1001, and 0b1010 are supported

Increment Event	Encoding	Notes
		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 a Gen 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	<p>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.</p> <p><b>Note:</b> There were no C steppings for the GPU, so this increment event is supported.</p> <p><b>Note:</b> A thread slot is considered to be "loaded" once the EU receives the transparent header. Hence, time spent transferring additional thread header data phases (e.g. push constants and vertex attribute data) will count towards thread occupancy.</p>

### Supported Coarse Event Filters

Coarse Event Filter	Encoding	Notes
No mask	0b0000	Never masks increment event.
Currently executing thread came from VS	0b0001	Masks increment event unless the FFID which dispatched the currently executing thread equals FFID of VS.
Currently executing thread came from HS	0b0010	Masks increment event unless the FFID which dispatched the currently executing thread equals FFID of HS.
Currently executing thread came from DS	0b0011	Masks increment event unless the FFID which dispatched the currently executing thread equals FFID of DS.
Currently executing thread came from GS	0b0100	Masks increment event unless the FFID which dispatched the currently executing thread equals FFID of GS.
Currently executing thread came from PS	0b0101	Masks increment event unless the FFID which dispatched the currently executing thread equals FFID of PS.
Currently executing thread came from TS	0b0110	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.
		Filter behaves unreliably when shader ISA uses 64-bit immediate values.
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**

## MI\_REPORT\_PERF\_COUNT

**MI\_REPORT\_PERF\_COUNT**