LogiCORE IP AXI Performance Monitor v4.0

Product Guide for Vivado Design Suite

PG037 June 19, 2013

Table of Contents

Introduction

The LogiCORE IP Advanced eXtensible Interface (AXI) Performance Monitor enables AXI system performance measurement for multiple slots (AXI4/AXI4-Stream). This core captures configurable real-time performance metrics for throughput and latency for connected AXI interfaces. In addition, the AXI Performance Monitor logs the AXI transactions, external system events and performs real-time profiling for software applications.

Features

- Connects as a 32-bit slave on AXI4-Lite interface
- Configurable number of (AXI4/ AXI4-Stream) monitor slots (up to eight)
- Flexible support for monitor slots with any data width, ID width and frequency
- Free running Global Clock Counter
- Supports AXI and external events logging
- Supports AXI and external events counting
- Supports external event triggering and cross probing between event counting and event logging

Additional features are listed in Chapter 1, Overview.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	Zynq®-7000, Virtex®-7, Kintex®-7, Artix®-7
Supported User Interfaces	AXI4-Stream, AXI4-Lite and AXI4
Resources	See Table 2-2, Table 2-3, and Table 2-4.
Provided with Core	
Design Files	Verilog
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	XDC
Simulation Model	None
Supported S/W Driver[2]	Standalone
Tested Design Flows[3]	
Design Entry	Vivado™ Design Suite
Simulation	Mentor Graphics Questa® SIM
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx @ www.xilinx.com/support	

Notes:

1. For a complete list of supported devices, see Vivado IP catalog.
2. Standalone driver details can be found in the SDK directory (<*install_directory*>/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from //wiki.xilinx.com.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The LogiCORE™ IP AXI Performance Monitor measures major performance metrics for the AMBA AXI system. The Performance Monitor measures bus latency of a specific master/ slave (AXI4/AXI4-Stream) in a system, the amount of memory traffic for specific durations, and other performance metrics. This core can also be used for real-time profiling for software applications.

The AXI Performance Monitor monitors all the signals on the connected AXI slot. The user configures the performance counter metric for the selected AXI slot. See Register Space in Chapter 2 for more details about the configuration registers.

The AXI Performance Monitor (APM) supports following two major functions to analyze the system behavior on AXI interfaces:

- Event Logging: The APM logs the specified AXI monitor slots' events (configured by user) and external system events coming in to the streaming FIFO. This data can be used by the system processor or external host application to reconstruct the AXI transaction for analyzing system behavior/performance.
- Event Counting : The APM supports monitoring AXI slots for counting events associated with AXI interface transaction and external events. The included event counters can be set, read by the software, and used to analyze and enhance the system performance. The core also has a global clock counter for real-time profiling for software applications.

The top-level block diagram of the AXI Performance Monitor is shown in Figure 1-1.

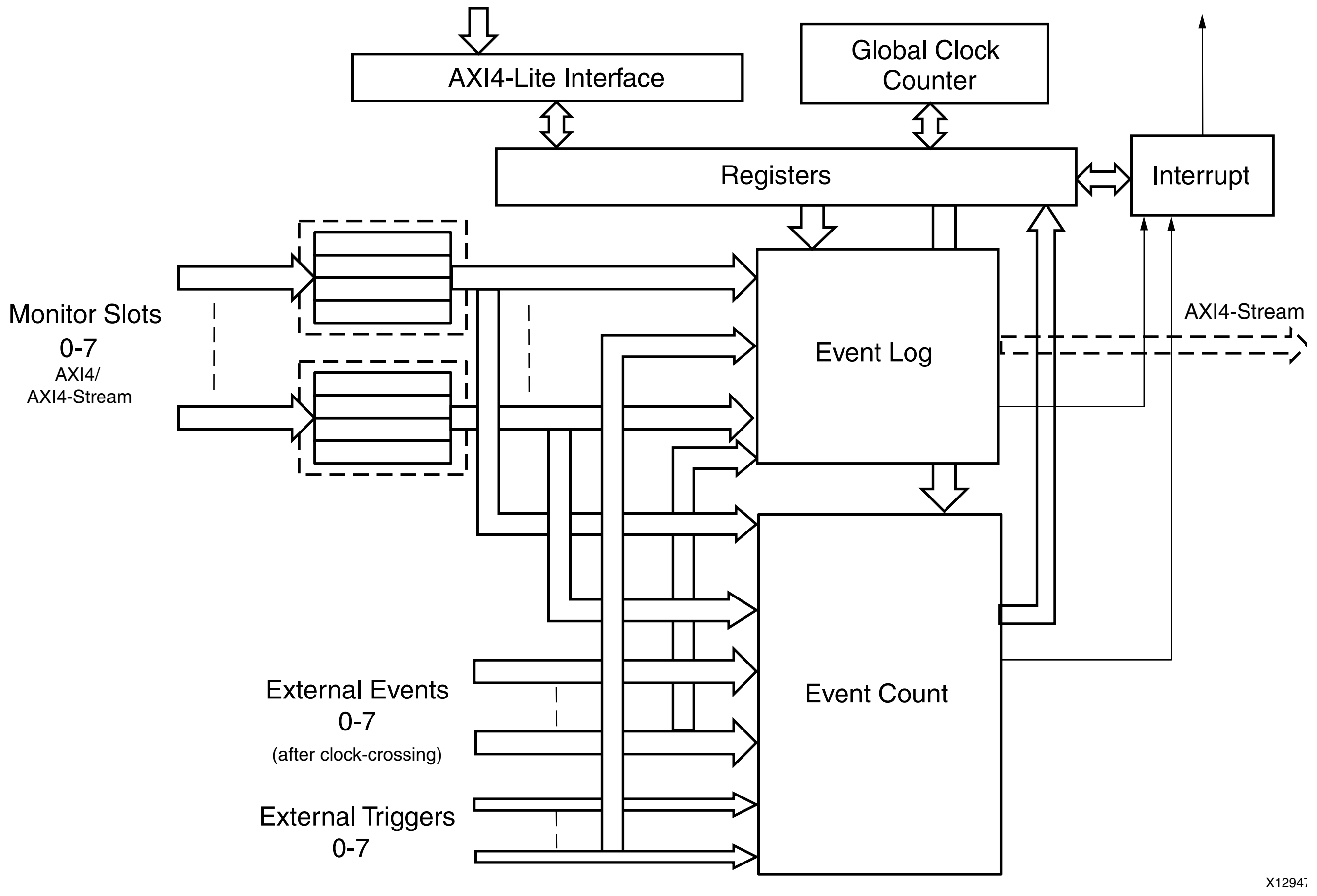


Figure 1-1: **Block Diagram of AXI Performance Monitor**

Event Logging

Event logging captures the specified AXI events, external events and the time stamp difference between two successive events into a streaming FIFO. The selection of events to be captured is done through the configuration register and by setting flags. The flags are set in the flag enable register.

The flags that can be selected for AXI4 event logging are given below.

- Last Read Flag
- First Read Flag
- Read Address Latch Flag
- Response Flag
- Last Write Flag
- First Write Flag
- Write Address Latch Flag

The following flags can be set for the AXI4-Stream event logging:

- Last Data Flag
- First Data Flag

Along with the flags, the event logging can track the AXI data for analyzing the transactions. The log data for AXI4 includes:

- AWID
- BID
- ARID
- RID
- AWLEN
- ARLEN

The log data for AXI4-Stream interfaces includes:

- TID
- TDEST
- TUSER

The packet format for the streaming FIFO is listed in Table 1-1 and Table 1-2. Table 1-2 shows the packet format for only two monitor slots. The width of the packet increases with the number of slots in a similar fashion. The AXI4 log data width is based on the user selection for enabling ID and Transaction Length for Log data when configuring the core. AXI4-Stream log data width is based on the user selection for enabling TID, TDEST and TUSER fields in log data.

The APM software-written data register allows an application to log user data, for example, software timestamp data along with the hardware timestamp difference. Either the software-written data register packet or the monitor log packet is sent over the Event Log streaming interface. When there is a write to the software-written data register, the software-written data register packet is sent; otherwise the monitor log packet is sent. The corresponding flags should be set for both the packets. If the software-written data register write and monitor slot events occur at the same time, first the software-written data register packet is sent and then the monitor log packet with a timestamp difference of '0' is sent. There is a Log ID bit in both the packets to distinguish the packet type (1 for software-written data register packet and 0 for monitor events packet).

Table 1-1: **SW Data Register Packet**

Name	Width	Is Valid	Description
Log ID	1		Packet type. • 1= SW data register
Time Stamp Difference	16		Contains the time stamp difference between the previous event (write to FIFO) and the current event.
Loop Event	1		Is '1' when the time stamp difference exceeds 2^16.
SW Data Register	32		Contains the software sync data register value when written to that register.

Table 1-2: **Monitor Log Packet**

Name	Width	Is Valid	Description
Log ID	1		Packet type. • 0=monitor slots log
Time Stamp Difference	16		Contains the time stamp difference between the previous event (write to FIFO) and the current event.
Loop Event	1		Is '1' when the time stamp difference exceeds 2^16.
External Event 0 Flags	3		External Event 0 start, stop and Event Flags are captured.
Slot 0 Flags	7	Slot 0 AXI Protocol = AXI4	Event Flags for Slot 0. Includes Last Read Flag, First Read Flag, Read Address Latch Flag, Response Flag, Last Write Flag, First Write Flag, Write Address Latch Flag.
Slot 0 Log Data	(Enable ID x Slot 0 ID Width x 4) + (Enable Length x 16)		AWID, BID, ARID, RID, AWLEN, ARLEN.
Slot 0 Flags	2	Slot 0 AXI Protocol = AXI4-Stream	Event Flags for Slot 0. Includes Last Data Flag, First Data Flag.
Slot 0 Log Data	(Enable TID x Slot 0 TID Width) + (Enable TDEST x Slot 0 TDEST Width) + (Enable TUSER x Slot 0 TUSER Width)		TID, TDEST, TUSER.
External Event 1 Flags	3	No of Monitor Slots > 1	External Event 1 start, stop and Event Flags are captured.

Table 1-2: **Monitor Log Packet *(Cont'd)***

Name	Width	Is Valid	Description
Slot 1 Flags	7	(Slot 1 AXI Protocol = AXI4) && (No of Monitor Slots > 1)	Event Flags for Slot 1. Includes Last Read Flag, First Read Flag, Read Address Latch Flag, Response Flag, Last Write Flag, First Write Flag, Write Address Latch Flag.
Slot 1 Log Data	(Enable ID x Slot 1 ID Width x 4) + (Enable Length x 16)		AWID, BID, ARID, RID, AWLEN, ARLEN.
Slot 1 Flags	2	(Slot 1 AXI Protocol = AXI4-Stream) && (No of Monitor Slots > 1)	Event Flags for Slot 1. Includes Last Data Flag, First Data Flag.
Slot 1 Log Data	(Enable TID x Slot 1 TID Width) + (Enable TDEST x Slot 1 TDEST Width) + (Enable TUSER x Slot 1 TUSER Width)		TID, TDEST, TUSER.

Event logging can be enabled through the Control register, or it can be auto-enabled by using the cross probing functionality of the core. Cross probing automatically starts event logging in any of the following conditions:

- When the global clock counter is overflow.
- When the sampled metric counter expires with a loaded value.
- When the metric counter values fall within the cut-off range set in the event logging cut-off registers.

Cross probing is enabled through the flag enable register bits [31:20].

Event Counting

The Event Count block diagram is shown in Figure 1-2.

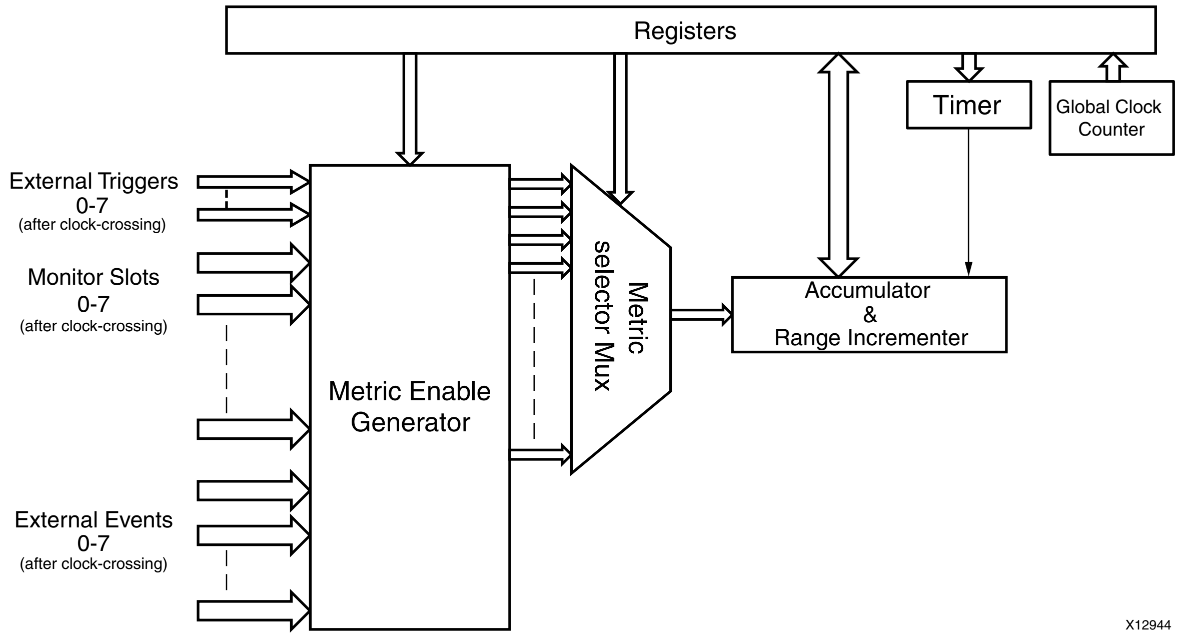

Registers
Timer
Global Clock Counter
External Triggers 0-7 (after clock-crossing)
Monitor Slots 0-7 (after clock-crossing)
External Events 0-7 (after clock-crossing)
Metric Enable Generator
Metric selector Mux
Accumulator & Range Incrementer
X12944

Figure 1-2: **Event Count Module**

Event Counting consists of a configurable number of Accumulator and Range Incrementer blocks (max of 10). All the events that can be counted are detected and given to the accumulator which gives the aggregate value. The selection of event to monitor is done through the metric selector registers associated with the counters. Event counting can be used for measuring events on AXI4/AXI4-Stream monitor slots or external event ports.

The metric counters can be enabled in two ways:

- Control Register Bit Enable: Enable each metric counter through the metric count enable bit in the control register.
- External Trigger Pulse: External trigger ports are available for each monitor AXI/AXI4-Stream slot. Enable the external trigger though the external trigger selection bit in the control register for the Metric Counter. External triggers are unused when the APM is used for external event counting.
  The external trigger pulse also needs the enable bit to be set in the control register. In addition, the external triggers can control the start of counters.

The Accumulator and Range Incrementer module is shown in Figure 1-3.

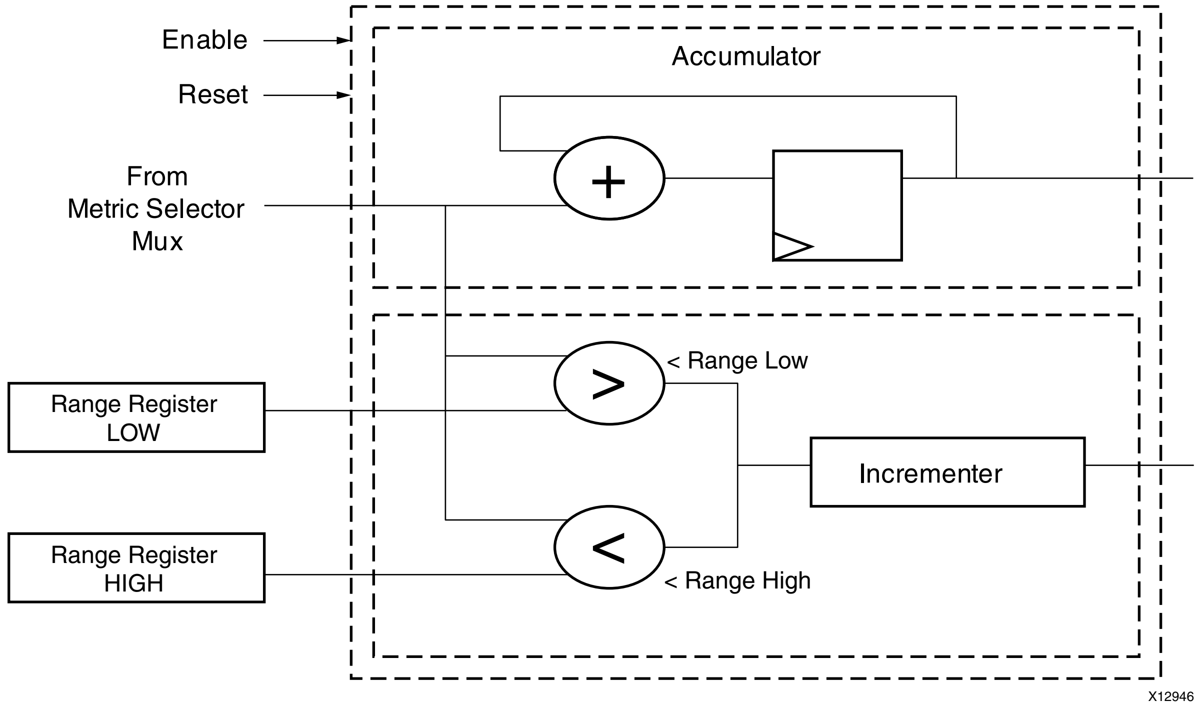


Figure 1-3: **Accumulator and Range Incrementer**

The Range Incrementer compares the event count with the low and high ranges from the Range register and increments the count by one if the value falls within the limits. The Accumulator and Incrementer values can be read through the AXI4-Lite interface. The Range Incrementer is useful in obtaining the read/write latency ranges.

The Metric Counter also consists of a timer which helps in sampling the accumulator and incrementer values at pre-configured time intervals. The metric counters can also be used to count the external events that fall between its corresponding start and stop signals.

The metrics are obtained for the selected agent/slot. This selection is also done through the metric selection registers.

Metrics computed for an AXI4 agent:

- **Write Request Count**: Total number of write requests by/to the agent.
- **Read Request Count**: Total number of read requests given by/to the agent.
- **Read Latency**: The time from the start of read address transaction to the beginning of the read data service for the configured AXI transaction ID.
- **Write Latency**: The time from the start of the write address transaction to the write response from the slave for the configured AXI transaction ID.
- **Write Byte Count**: Total number of bytes written by/to the agent. This metric is helpful when calculating the throughput of the system.

- **Read Byte Count**: Total number of bytes read from/by the agent.
- **Slave Write Idle Cycle Count**: Number of idle cycles caused by the slave during write transactions to the slave. The slave write idle cycles are the number of clocks between `WVALID` assertion and `WREADY` assertion.
- **Master Read Idle Cycle Count**: Number of idle cycles caused by the master during read transactions to the slave. The master read idle cycles are the number of clocks between `RVALID` assertion and `RREADY` assertion.

Metrics computed for an AXI4-Stream agent:

- **Transfer Cycle Count**: Total number of packets transferred.
- **Data Byte Count**: Total number of data bytes transferred. This metric helps in calculating the throughput of the system.
- **Position Byte Count**: Total number of position bytes transferred.
- **Null Byte Count**: Total number of null bytes transferred.
- **Packet Count**: Total number of packets transferred.
- **Idle Count**: Number of idle cycles caused by the AXI4-Stream interface.

System-level metrics like write data throughput, read data throughput, and interconnect read latency can be computed by obtaining all metrics for all the agents in the system.

Applications

The AXI Performance Monitor has the following applications:

- Studying the latencies involved for any AXI-based slave, like a memory controller, and tuning the core.
- Comparing different applications by facilitating benchmarking.
- Debugging a system, for example, counting the responses against requests.
- Obtaining charts like latency distribution, throughput, burst distribution and others for a slave and across the system.
- Obtaining the system-level metrics like write throughput, read throughput, average interconnect read latency and others.
- Obtaining the run time of an application and optimizing the software.
- Analyzing the latencies involved in transactions by identifying the agent causing more idle cycles in the transactions.
- Comparing two similar AXI agents.

- Counting the interested external events (other than AXI) like FIFO overflow/underflow, interrupts, and others.
- Logging the important or interested events on the monitor slots, reconstruct and analyze the behavior/performance.

Unsupported Features

The AXI Performance Monitor has the following known limitations:

- Bus contention metrics are not computed.
- Due to logic constraints, the core does not provide all the metrics for all the agents in the system in a single run of the application.
- Supports a maximum outstanding transaction depth of 32. The write/read latency metrics are affected by the outstanding transactions. The core captures the read/write latency of ID transactions one at a time.
- Does not support interleaved transactions.

Licensing

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx Vivado Design Suite under the terms of the Xilinx End User License.

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

This chapter contains resource usage data, signal descriptions and details about the registers.

Performance

Performance characterization of this core has been done using margin system methodology. The details of the margin system characterization methodology are available in the Vivado Design Suite User Guide: Designing with IP (UG896).

Maximum Frequencies

Table 2-1 lists the maximum frequencies for this core.

Note: Maximum frequency numbers for Zynq-7000 devices are expected to be similar to 7 series device numbers.

Table 2-1: **Maximum Frequencies**

Family	Speed Grade	F_{MAX} (MHz)	
		AXI4	AXI4-Lite
Virtex-7	-1	200	180
Kintex-7		180	150
Artix-7		150	120
Virtex-7	-2	220	180
Kintex-7		200	150
Artix-7		160	120
Virtex-7	-3	250	180
Kintex-7		220	150
Artix-7		170	120

Latency

The latency is the worst time the performance monitor requires for providing valid metric counters data through registers.

When the monitor clock and core clock are same, the latency is six clocks. When the monitor clock and core clock are asynchronous, Asynchronous FIFO latency is added to the standard latency, and so the effective latency is four monitor clocks and 13 core clocks.

Resource Utilization

Resources required for the AXI Performance Monitor core have been estimated for the 7 series FPGAs (Table 2-2 to Table 2-4). They are derived from post-synthesis reports, and might change during MAP and PAR.

Note: Resource numbers for Zynq-7000 devices are expected to be similar to 7 series device numbers.

The AXI Performance Monitor resource utilization for various parameter combinations measured with a Virtex-7 FPGA as the target device are detailed in Table 2-2.

Table 2-2: **Resource Estimates for Virtex-7 FPGAs**

Parameter Values			Device Resource		
Monitor Slots	**Counters**	**Event Logging**	**Slices**	**Slice Flip-Flops**	**LUTs**
1	1	0	500	1650	1123
1	5	0	1044	2347	2853
1	5	1	1310	2799	3159
4	10	0	2766	5809	7065
4	10	1	3003	6981	7554
8	10	0	4397	10122	10465
8	10	1	5291	12487	11561

The AXI Performance Monitor resource utilization for various parameter combinations measured with a Kintex-7 FPGA as the target device are detailed in Table 2-3.

Table 2-3: **Resource Estimates for Kintex-7 FPGAs**

Parameter Values			Device Resource		
Monitor Slots	**Counters**	**Event Logging**	**Slices**	**Slice Flip-Flops**	**LUTs**
1	1	0	505	1370	1334
1	5	0	1076	2347	2853
1	5	1	1195	2799	3159
4	10	0	2617	5809	7063
4	10	1	2971	6981	7555

Table 2-3: **Resource Estimates for Kintex-7 FPGAs *(Cont'd)***

Parameter Values			Device Resource		
Monitor Slots	**Counters**	**Event Logging**	**Slices**	**Slice Flip-Flops**	**LUTs**
8	10	0	4527	8782	11579
8	10	1	5132	10917	12349

The AXI Performance Monitor resource utilization for various parameter combinations measured with a Artix-7 FPGA as the target device are detailed in Table 2-4.

Table 2-4: **Resource Estimates for Artix-7 FPGAs**

Parameter Values			Device Resource		
Monitor Slots	**Counters**	**Event Logging**	**Slices**	**Slice Flip-Flops**	**LUTs**
1	1	0	475	1349	1260
1	5	0	1024	2326	2894
1	5	1	1254	2799	3278
4	10	0	2590	5809	7063
4	10	1	2979	6981	7554
8	10	0	4348	8782	11670
8	10	1	4968	10917	12349

Port Descriptions

Table 2-5 lists the AXI Performance Monitor signals. The I/O signals include AXI interface signals through which the IP is configured and registers are read, and Monitor slot I/O signals to which the agents (masters/slaves) that need to be monitored are connected.

Table 2-5: **AXI Performance Monitor I/O Signals**

Signal Name	Interface	I/O	Initial State	Description
System Interface				
core_aclk	System	I	-	Core Clock. This clock should be the fastest clock among all clocks of AXI Performance Monitor Core.
core_aresetn	System	I	-	Core Reset. Active-Low.
capture_event	System	I	-	Event Capture.
reset_event	System	I	-	Event Reset.

Table 2-5: **AXI Performance Monitor I/O Signals *(Cont'd)***

Signal Name	Interface	I/O	Initial State	Description
interrupt	System	O	0x0	System Interrupt Output.
AXI4-Lite Slave Interface				
s_axi_aclk	System	I		AXI Clock.
s_axi_aresetn	System	I		AXI Reset. Active-Low.
s_axi_*	S_AXI	-	-	See Appendix A of the *AXI Reference Guide* (UG761) for description of AXI4-Lite Signals.
Monitor Slot n (0 to 7) - AXI4(1)				
slot_n_axi_aclk	System	I	-	AXI Clock.
slot_n_axi_aresetn	System	I	-	AXI Reset. Active Low.
slot_n_axi_*	SLOT_n_AXI	I	-	See Appendix A of the AXI Reference Guide (UG761) for description of AXI4 Signals.
Monitor Slot n (0 to 7) - AXI4-Stream(1)				
slot_n_axis_aclk	System	I	-	AXI Clock.
slot_n_axis_aresetn	System	I	-	AXI Reset. Active-Low.
slot_n_axis_tvalid	SLOT_n_AXIS	I	-	Indicates that master is driving valid transfer.
slot_n_axis_tready	SLOT_n_AXIS	I	-	Indicates that slave can access transfer in current cycle.
slot_n_axis_tdata	SLOT_n_AXIS	I	-	Streaming data payload.
slot_n_axis_tstrb	SLOT_n_AXIS	I	-	Byte qualifier (data byte or a position byte).
slot_n4_axis_tkeep	SLOT_n_AXIS	I	-	Byte qualifier (data byte or a null byte).
slot_n_axis_tlast	SLOT_n_AXIS	I	-	Indicates last data beat of a packet.
slot_n_axis_tid	SLOT_n_AXIS	I	-	Data stream identifier.
slot_n_axis_tdest	SLOT_n_AXIS	I	-	Indicates routing information for data stream.
slot_n_axis_tuser	SLOT_n_AXIS	I	-	Indicates user defined sideband signals.
External Trigger Interface (0-7)(1)				
slot_n_ext_trig	System	I	-	External trigger start pulse for Slot n.
slot_n_ext_trig_stop	System	I	-	External trigger stop pulse for Slot n.

Table 2-5: **AXI Performance Monitor I/O Signals *(Cont'd)***

Signal Name	Interface	I/O	Initial State	Description
External Events Interface (0-7)[1]				
ext_clk_n	System	I	-	External Event (n) interface clock.
ext_rstn_n	System	I	-	External Event (n) interface reset. Active low.
ext_event_n_cnt_start	System	I	-	External Event (n) Count start signal.
ext_event_n_cnt_stop	System	I	-	External Event (n) Count stop signal.
ext_event_n	System	I	-	External Event.
Event Log Streaming Interface				
m_axis_aclk	System	I	-	AXI streaming clock.
m_axis_aresetn	System	I	-	AXI Streaming Reset. Active-Low.
m_axis_tdata	M_AXIS	O	0x0	Streaming data.
m_axis_tstrb	M_AXIS	O	0xF	Byte qualifier for streaming data.
m_axis_tvalid	M_AXIS	O	0x0	Indicates that the master is driving valid transfer.
m_axis_tid	M_AXIS	O	0x0	Data stream identifier.
m_axis_tready	M_AXIS	I	-	Indicates that the slave can accept a transfer in the current cycle.

1. AXI Performance Monitor has eight configurable monitor slots and *n* can vary from 0 to 7.

Register Space

This section details the registers and reset values of the AXI Performance Monitor core. Table 2-6 shows all the AXI Performance Monitor registers and addresses.

Table 2-6: **Core Registers**

Address Offset	Register Name	Description
Global Clock Counter Registers		
0x0000	Global Clock Counter	Higher 32-bit data of the Global Clock Counter Register
0x0004	Global Clock Counter	Lower 32-bit data of the Global Clock Counter Register

Table 2-6: **Core Registers *(Cont'd)***

Address Offset	Register Name	Description
Sample Interval Registers		
0x0020	Sample Interval MSB	Higher 32-bit data of Sample Interval Register
0x0024	Sample Interval LSB	Lower 32-bit data of Sample Interval Register
0x0028	Sample Interval Control Register	Sample Interval Control Register
Interrupt Registers		
0x0030	Global Interrupt Enable Register	Global Interrupt Enable Register
0x0034	Interrupt Enable Register	Interrupt Enable Register
0x0038	Interrupt Status Register	Interrupt Status Register
Metric Selector Registers		
0x0044	Metric Selector Register 0	Metric Selector for Metric Counters 0, 1, 2 and 3
0x0048	Metric Selector Register 1	Metric Selector for Metric Counters 4, 5, 6 and 7
0x004C	Metric Selector Register 2	Metric Selector for Metric Counter, 8 and 9
Metric Counters (Accumulators, Incrementers and Range Registers)		
0x0100	Metric Counter 0	Metric Counter 0 Register
0x0104	Incrementer 0	Incrementer 0 Register
0x0108	Range Register 0	Low & High Ranges for Incrementer 0
0x010C	Metric Count Log Enable Register 0	Metric Count Log Enable Register 0
0x0110	Metric Counter 1	Metric Counter 1 Register
0x0114	Incrementer 1	Incrementer 1 Register
0x0118	Range Register 1	Low & High Ranges for Incrementer 1
0x011C	Metric Count Log Enable Register 1	Metric count log enable register1
0x0120	Metric Counter 2	Metric Counter 2 Register
0x0124	Incrementer 2	Incrementer 2 Register
0x0128	Range Register 2	Low & High Ranges for Incrementer 2
0x012C	Metric Count Log Enable Register 2	Metric Count Log Enable Register 2
0x0130	Metric Counter 3	Metric Counter 3 Register
0x0134	Incrementer 3	Incrementer 3 Register
0x0138	Range Register 3	Low & High Ranges for Incrementer 3
0x013C	Metric Count Log Enable Register 3	Metric Count Log Enable Register 3
0x0140	Metric Counter 4	Metric Counter 4 Register

Table 2-6: **Core Registers *(Cont'd)***

Address Offset	Register Name	Description
0x0144	Incrementer 4	Incrementer 4 Register
0x0148	Range Register 4	Low & High Ranges for Incrementer 4
0x014C	Metric Count Log Enable Register 4	Metric Count Log Enable Register4
0x0150	Metric Counter 5	Metric Counter 5 Register
0x0154	Incrementer 5	Incrementer 5 Register
0x0158	Range Register 5	Low & High Ranges for Incrementer 5
0x015C	Metric Count Log Enable Register 5	Metric Count Log Enable Register 5
0x0160	Metric Counter 6	Metric Counter 6 Register
0x0164	Incrementer 6	Incrementer 6 Register
0x0168	Range Register 6	Low & High Ranges for Incrementer 6
0x016C	Metric Count Log Enable Register 6	Metric Count Log Enable Register 6
0x0170	Metric Counter 7	Metric Counter 7 Register
0x0174	Incrementer 7	Incrementer 7 Register
0x0178	Range Register 7	Low & High Ranges for Incrementer 7
0x017C	Metric Count Log Enable Register 7	Metric Count Log Enable Register 7
0x0180	Metric Counter 8	Metric Counter 8 Register
0x0184	Incrementer 8	Incrementer 8 Register
0x0188	Range Register 8	Low & High Ranges for Incrementer 8
0x018C	Metric Count Log Enable Register 8	Metric Count Log Enable Register 8
0x0190	Metric Counter 9	Metric Counter 9 Register
0x0194	Incrementer 9	Incrementer 9 Register
0x0198	Range Register 9	Low & High Ranges for Incrementer 9
0x019C	Metric Count Log Enable Register 9	Metric Count Log Enable Register 9
Sampled Metric Counter Registers		
0x0200	Sampled Metric Counter 0	Sampled Metric Counter 0 Register
0x0204	Sampled Incrementer 0	Sampled Incrementer 0 Register
0x0210	Sampled Metric Counter 1	Sampled Metric Counter 1 Register
0x0214	Sampled Incrementer 1	Sampled Incrementer 1 Register
0x0220	Sampled Metric Counter 2	Sampled Metric Counter 2 Register
0x0224	Sampled Incrementer 2	Sampled Incrementer 2 Register
0x0230	Sampled Metric Counter 3	Sampled Metric Counter 3 Register

Table 2-6: **Core Registers *(Cont'd)***

Address Offset	Register Name	Description
0x0234	Sampled Incrementer 3	Sampled Incrementer 3 Register
0x0240	Sampled Metric Counter 4	Sampled Metric Counter 4 Register
0x0244	Sampled Incrementer 4	Sampled Incrementer 4 Register
0x0250	Sampled Metric Counter 5	Sampled Metric Counter 5 Register
0x0254	Sampled Incrementer 5	Sampled Incrementer 5 Register
0x0260	Sampled Metric Counter 6	Sampled Metric Counter 6 Register
0x0264	Sampled Incrementer 6	Sampled Incrementer 6 Register
0x0270	Sampled Metric Counter 7	Sampled Metric Counter 7 Register
0x0274	Sampled Incrementer 7	Sampled Incrementer 7 Register
0x0280	Sampled Metric Counter 8	Sampled Metric Counter 8 Register
0x0284	Sampled Incrementer 8	Sampled Incrementer 8 Register
0x0290	Sampled Metric Counter 9	Sampled Metric Counter 9 Register
0x0294	Sampled Incrementer 9	Sampled Incrementer 9 Register
Control Register		
0x0300	Control Register	Control Register
0x0304	Latency ID Register	Latency ID Register
Event Log Registers		
0x0400	Flag Enable Register	Flag Enable Register
0x0404	Software-written Data Register	Software-written Data Register

Global Clock Count Register (GCCR)

The Global Clock Count is a 32/64 read register. This register holds the total number of simulation cycles. The bit definition and accessibility of this register are shown in Table 2-7.

Table 2-7: **Global Clock Count Register Bit Definitions (0x0004)**

Bits	Name	Access	Reset Value	Description
63-32	Global Clock Count	Read	0x00	Higher 32-bits of Global Clock Count Register
31-0	Global Clock Count	Read	0x00	Lower 32-bits of Global Clock Count Register

Notes:

1. Because the S_AXI data width is 32 bits, the register can be read by a read transaction to two locations: (0x0000) and (0x0004).
2. If Global Count Width is 32, only the lower 32 bits of the register are valid.
3. If Global Count Width is 64, then all 64 bits of the register are valid.

Sample Interval Register (SIR)

The Sample Interval is a 32/64-bit read/write register. This register holds the load value for the sample interval down counter used in generating a capture event for capturing the metric counters into Sampled Metric Counter registers. The bit definition and accessibility of this register is shown in Table 2-8.

Table 2-8: **Sample Interval Register Bit Definitions (0x0024)**

Bits	Name	Access	Reset Value	Description
63-32	Sample Interval	Read / Write	0x00	Higher 32-bits of Sample Interval Register
31-0	Sample Interval	Read / Write	0x00	Lower 32-bits of Sample Interval Register

Notes:

1. Because the S_AXI data width is 32 bits, the register can be read by a read transaction to two locations: (C_ 0x0020) and (C_ 0x0024).
2. If the Sample Interval Counter Width is 32, only the lower 32-bits of the register are valid.
3. If the Sample Interval Counter Width is 64, then higher 32-bits of the register are also valid.

Sample Interval Control Register (SICR)

The Sample Interval Control Register is a 32-bit register. This register is used to control the sample interval down counter inside the monitor. The bit definition and accessibility of this register is shown in Table 2-9.

Table 2-9: **Sample Interval Control Register Bit Definitions (0x0028)**

Bits	Name	Access	Reset Value	Description
31-9	Reserved	N/A	N/A	Reserved
8	Metric_Counters_Reset_Enable	Read/Write	0	1: Metric Counters are reset when sample interval counter lapses.
7-2	Reserved	N/A	N/A	Reserved
1	Load	Read/Write	0	1: Loads the Sample Interval register value into the Sample Interval Counter.
0	Enable	Read/Write	0	1: Enables the down counter. Before enabling, the counter should be loaded with the sample Interval Register value.

Global Interrupt Enable Register (GIER)

The Global Interrupt Enable Register provides the master enable/disable for the interrupt output to the processor. This is a single read/write register.

Table 2-10: **Global Interrupt Enable Register Bit Definitions (0x0030)**

Bits	Name	Access	Reset Value	Description
31-1	Reserved	N/A	N/A	Reserved
0	GIE	Read/Write	0	Master enable for the device interrupt output to the system interrupt controller: • 1: Enabled • 0: Disabled

Interrupt Enable Register (IER)

This is a read/write register. Writing a '1' to a bit in this register enables the corresponding ISR bit to cause assertion of the Interrupt. An IER bit set to '0' does not inhibit an interrupt condition for being captured, just reported. Writing a '0' to a bit disables, or masks, will disable the generation of interrupt output for corresponding interrupt signal.

Table 2-11: **Interrupt Enable Register Bit Definitions (0x0034)**

Bits	Name	Access	Reset Value	Description
31-13	Reserved	N/A	N/A	Reserved
12	Metric Counter 9 Overflow Interrupt Enable	Read/Write	0	Metric Counter 9 Overflow Interrupt • 1: Enabled • 0: Disabled
11	Metric Counter 8 Overflow Interrupt Enable	Read/Write	0	Metric Counter 8 Overflow Interrupt • 1: Enabled • 0: Disabled
10	Metric Counter 7 Overflow Interrupt Enable	Read/Write	0	Metric Counter 7 Overflow Interrupt • 1: Enabled • 0: Disabled
9	Metric Counter 6 Overflow Interrupt Enable	Read/Write	0	Metric Counter 6 Overflow Interrupt • 1: Enabled • 0: Disabled
8	Metric Counter 5 Overflow Interrupt Enable	Read/Write	0	Metric Counter 5 Overflow Interrupt • 1: Enabled • 0: Disabled
7	Metric Counter 4 Overflow Interrupt Enable	Read/Write	0	Metric Counter 4 Overflow Interrupt • 1: Enabled • 0: Disabled

Table 2-11: **Interrupt Enable Register Bit Definitions (0x0034) *(Cont'd)***

Bits	Name	Access	Reset Value	Description
6	Metric Counter 3 Overflow Interrupt Enable	Read/Write	0	Metric Counter 3 Overflow Interrupt • 1: Enabled • 0: Disabled
5	Metric Counter 2 Overflow Interrupt Enable	Read/Write	0	Metric Counter 2 Overflow Interrupt • 1: Enabled • 0: Disabled
4	Metric Counter 1 Overflow Interrupt Enable	Read/Write	0	Metric Counter 1 Overflow Interrupt • 1: Enabled • 0: Disabled
3	Metric Counter 0 Overflow Interrupt Enable	Read/Write	0	Metric Counter 0 Overflow Interrupt • 1: Enabled • 0: Disabled
2	Event Log FIFO full Interrupt Enable	Read/Write	0	Event Log FIFO full Interrupt • 1: Enabled • 0: Disabled
1	Sample Interval Counter Overflow Interrupt Enable	Read/Write	0	Sample Interval Counter Overflow Interrupt • 1: Enabled • 0: Disabled
0	Global Clock Counter Overflow Interrupt Enable	Read/Write	0	Global Clock Counter Overflow Interrupt • 1: Enabled • 0: Disabled

Interrupt Status Register (ISR)

When read, the contents of this register indicate the presence or absence of an active interrupt signal. Each bit in this register that is set to a '1' indicates an active interrupt signal. Bits that are '0' are not active. The bits in the ISR are independent of the interrupt enable bits in the IER. The interrupt can be cleared by writing '1' to the corresponding bit in the Interrupt Status registers.

Table 2-12: **Interrupt Status Register Bit Definitions (0x0038)**

Bits	Name	Access	Reset Value	Description
31-13	Reserved	N/A	N/A	Reserved
12	Metric Counter 9 Overflow Interrupt	Read/Write	0	Metric Counter 9 Overflow Interrupt • 1: Active • 0: Not Active

Table 2-12: **Interrupt Status Register Bit Definitions (0x0038)** ***(Cont'd)***

Bits	Name	Access	Reset Value	Description
11	Metric Counter 8 Overflow Interrupt	Read/Write	0	Metric Counter 8 Overflow Interrupt • 1: Active • 0: Not Active
10	Metric Counter 7 Overflow Interrupt	Read/Write	0	Metric Counter 7 Overflow Interrupt • 1: Active • 0: Not Active
9	Metric Counter 6 Overflow Interrupt	Read/Write	0	Metric Counter 6 Overflow Interrupt • 1: Active • 0: Not Active
8	Metric Counter 5 Overflow Interrupt	Read/Write	0	Metric Counter 5 Overflow Interrupt • 1: Active • 0: Not Active
7	Metric Counter 4 Overflow Interrupt	Read/Write	0	Metric Counter 4 Overflow Interrupt • 1: Active • 0: Not Active
6	Metric Counter 3 Overflow Interrupt	Read/Write	0	Metric Counter 3 Overflow Interrupt • 1: Active • 0: Not Active
5	Metric Counter 2 Overflow Interrupt	Read/Write	0	Metric Counter 2 Overflow Interrupt • 1: Active • 0: Not Active
4	Metric Counter 1 Overflow Interrupt	Read/Write	0	Metric Counter 1 Overflow Interrupt • 1: Active • 0: Not Active
3	Metric Counter 0 Overflow Interrupt	Read/Write	0	Metric Counter 0 Overflow Interrupt • 1: Active • 0: Not Active
2	Event Log FIFO full Interrupt	Read/Write	0	Event Log FIFO full Interrupt • 1: Active • 0: Not Active

Table 2-12: **Interrupt Status Register Bit Definitions (0x0038)** ***(Cont'd)***

Bits	Name	Access	Reset Value	Description
1	Sample Interval Counter Overflow Interrupt	Read/Write	0	Sample Interval Counter Overflow Interrupt • 1: Active • 0: Not Active
0	Global Clock Counter Overflow Interrupt	Read/Write	0	Global Clock Counter Overflow Interrupt • 1: Active • 0: Not Active

Metric Selector Register 0 (MSR-0)

The Metric Selector Register 0 is a 32-bit register. This register is used to select the kind of metrics computed by the first four counters. The bit definition and accessibility of this register are shown in Table 2-13.

Table 2-13: **Metric Selector Register 0 Bit Definitions (0x0044)**

Bits	Name	Access	Reset Value	Description
31-29	Metric Counter 3 Slot ID	Read/Write	X00	Selects the Slot for the metric computed by Counter3
28-24	Metric Selector for Counter 3	Read/Write	x00	Selects the kind of metrics computed by Counter 3.
23-21	Metric Counter 2 Slot ID	Read/Write	X00	Selects the Slot for the metric computed by Counter2
20-16	Metric Selector for Counter 2	Read/Write	X00	Selects the kind of metrics computed by Counter 2.
15-13	Metric Counter 1 Slot ID	Read/Write	X00	Selects the Slot for the metric computed by Counter1
12-8	Metric Selector for Counter 1	Read/Write	X00	Selects the kind of metrics computed by Counter 1.
7-5	Metric Counter 0 Slot ID	Read/Write	X00	Selects the Slot for the metric computed by Counter0
4-0	Metric Selector for Counter 0	Read/Write	X00	Selects the kind of metrics computed by Counter 0.

Notes:

1. The number of Metric Counters that exist in the system is based on the No of Event Counters parameter.
2. The bits are reserved if the counter is not enabled through the No of Event Counters parameter.

Metric Selector Register 1 (MSR-1)

The Metric Selector Register 1 is a 32-bit register. This register is used to select the kind of metrics computed by counters 4 to 7. The bit definition and accessibility of this register are shown in Table 2-14.

Table 2-14: **Metric Selector Register 1 Bit Definitions (0x0048)**

Bits	Name	Access	Reset Value	Description
31-29	Metric Counter 7 Slot ID	Read/Write	X00	Selects the Slot for the metric computed by Counter7
28-24	Metric Selector for Counter 7	Read/Write	X00	Selects the kind of metrics computed by Counter 7.
23-21	Metric Counter 6 Slot ID	Read/Write	X00	Selects the Slot for the metric computed by Counter6
20-16	Metric Selector for Counter 6	Read/Write	X00	Selects the kind of metrics computed by Counter 6.
15-13	Metric Counter 5 Slot ID	Read/Write	X00	Selects the Slot for the metric computed by Counter5
12-8	Metric Selector for Counter 5	Read/Write	X00	Selects the kind of metrics computed by Counter 5.
7-5	Metric Counter 4 Slot ID	Read/Write	X00	Selects the Slot for the metric computed by Counter4
4-0	Metric Selector for Counter 4	Read/Write	X00	Selects the kind of metrics computed by Counter 4.

Notes:

1. The number of Metric Counters that exist in the system is based on the No of Event Counters parameter.
2. The bits are reserved if the counter is not enabled through the No of Event Counters parameter.

Metric Selector Register 2 (MSR-2)

The Metric Selector Register 2 is a 32-bit register. This register is used to select the kind of metrics computed by counters 8 to 9. The bit definition and accessibility of this register are shown in Table 2-15.

Table 2-15: **Metric Selector Register 2 Bit Definitions (0x004C)**

Bits	Name	Access	Reset Value	Description
31-16	Reserved	N/A	N/A	Reserved
15-13	Metric Counter 9 Slot ID	Read/Write	X00	Selects the Slot for the metric computed by Counter9
12-8	Metric Selector for Counter 9	Read/Write	X00	Selects the kind of metrics computed by Counter 9.
7-5	Metric Counter 8 Slot ID	Read/Write	X00	Selects the Slot for the metric computed by Counter8
4-0	Metric Selector for Counter 8	Read/Write	X00	Selects the kind of metrics computed by Counter 8.

Notes:

1. The number of Metric Counters that exist in the system is based on the No of Event Counters parameter.
2. The bits are reserved if the counter is not enabled through the No of Event Counters parameter.

Metric Selector values and the corresponding metric that is computed are described in Table 2-16 and Table 2-17.

Table 2-16: **Metric Descriptions**

Metric Selector Value	Metric	Description
Metrics Computed for AXI4 Agent		
0	Write Transaction Count	Number of write transactions by/to a particular master/ slave.
1	Read Transaction Count	Number of read transactions by/to a particular master/ slave.
2	Write Byte Count	Number of bytes written by/to a particular master/slave.
3	Read Byte Count	Number of bytes read from/by a particular slave/master.
4	Write Beat Count	Number of beats written by/to a particular master/slave.
5	Total Read Latency	Used with Num_Rd_Reqs (Read Transaction Count) to compute the Average Read Latency. This metric is for the selected ID transactions.
6	Total Write Latency	Used with Num_Wr_Reqs (Write Transaction Count) to determine the Average Write Latency. This metric is for the selected ID transactions.
7	Slv_Wr_Idle_Cnt	Number of idle cycles caused by the slave during a Write transaction.
8	Mst_Rd_Idle_Cnt	Number of idle cycles caused by the master during a read transaction.
9	Num_BValids	Number of BValids given by a slave to the master. This count helps in checking the responses against the number of requests given.
10	Num_WLasts	Number of WLasts given by the master. This count should exactly match the number of requests given by the master. This helps in debugging of the system.
11	Num_RLasts	Number of RLasts given by the slave to the master. This count helps in checking the responses against requests. This count should exactly match the number of requests given by the master.
12	Minimum Write Latency	Minimum write latency number for the selected ID transactions. The default minimum write latency provided by the core is 0xFFFFFFFF.
13	Maximum Write Latency	Maximum write latency number for the selected ID transactions.
14	Minimum Read Latency	Minimum Read Latency number for the selected ID transactions. The default minimum read latency provided by the core is 0xFFFFFFFF.
15	Maximum Read Latency	Maximum Read latency number for the selected ID transactions.
Metrics Computed for AXI4-Stream Agent		

Table 2-16: **Metric Descriptions *(Cont'd)***

Metric Selector Value	Metric	Description
16	Transfer Cycle Count	Gives the total number of cycles the data is transferred.
17	Packet Count	Gives the total number of packets transferred.
18	Data Byte Count	Gives the total number of data bytes transferred.
19	Position Byte Count	Gives the total number of position bytes transferred.
20	Null Byte Count	Gives the total number of null bytes transferred.
21	Slv_Idle_Cnt	Gives the number of idle cycles caused by the slave.
22	Mst_Idle_Cnt	Gives the number of idle cycles caused by the master.
External Events		
30	External event count	Gives the count for external event. Slot ID gives which external event count.

Table 2-17: **Metric Slot Descriptions**

Metric Slot ID Value	Slot	Description
0	Slot 0	Slot0 metric. When metric is External Event Count, Ext Event0 count is captured in the metric counter.
1	Slot 1	Slot1 metric. When metric is External Event Count, Ext Event1 count is captured in the metric counter.
2	Slot 2	Slot2 metric. When metric is External Event Count, Ext Event2 count is captured in the metric counter.
3	Slot 3	Slot3 metric. When metric is External Event Count, Ext Event3 count is captured in the metric counter.
4	Slot 4	Slot4 metric. When metric is External Event Count, Ext Event4 count is captured in the metric counter.
5	Slot 5	Slot5 metric. When metric is External Event Count, Ext Event5 count is captured in the metric counter.
6	Slot 6	Slot6 metric. When metric is External Event Count, Ext Event6 count is captured in the metric counter.
7	Slot 7	Slot7 metric. When metric is External Event Count, Ext Event7 count is captured in the metric counter

Metric Counter x Registers (MCR)

The Metric Counter Registers are of 32-bit wide. These contain the accumulated value of the events that are chosen. The bit definition and accessibility of this register are shown in Table 2-18.

Table 2-18: **Metric Counter Register Bit Definitions (0x0100)**

Bits	Name	Access	Reset Value	Description
31-0	Metric Counter (Accumulator)	Read	0	Contains the accumulated value of the event selected

Notes:
1. Number of Metric Counters that exist in the system is based on No of Event Counters options set through GUI.
2. x varies from 0 to No of Event Counters-1.
3. Refer to the register space table for addresses of all the metric counters.

Incrementer x Registers (IR)

The Incrementer Registers are of 32-bit wide. These contain the value that the number of times a selected event fallen in the chosen range for Incrementer. The bit definition and accessibility of this register are shown in Table 2-19. These registers are useful for read latency and write latency to obtain the latency distribution in a given range. For other metrics, the metric counter value suffice making this register redundant.

Table 2-19: **Incrementer Register Bit Definitions (0x0104)**

Bits	Name	Access	Reset Value	Description
31-0	Incrementer	Read	0	Contains the value of the Incrementer

Notes:
1. Number of Incrementers that exist in the system is based on No of Event Counter parameter set in GUI.
2. x varies from 0 to No of Event Counters-1.
3. Refer to the register space table for addresses of all the Incrementer Registers.

Range Register x (RR)

The Range Registers are of 32-bit wide. These registers help in incrementing the incrementer when the selected event falls in the range specified here. The bit definition and accessibility of this register are shown in Table 2-20.

Table 2-20: **Range Register Bit Definitions (0x0108)**

Bits	Name	Access	Reset Value	Description
31-16	Range HIGH	Read / Write	0x00	Contains the HIGHER limit of a range
15-0	Range LOW	Read / Write	0x00	Contains the LOWER limit of a range

Notes:
1. Number of Range Registers that exist in the system is based on No of Event Counters parameter set in the GUI.
2. x varies from 0 to No Event Counters-1.
3. Refer to the register space table for addresses of all the Range Registers.

Metric Count Log Enable Register *x* (MCLER)

The Metric Count Log Enable registers are 32 bits wide. These registers help to compare the metric counters and enable the event log if the metric count value is greater than or equal to the corresponding register value. The bit definition and accessibility of this register are shown in Table 2-21.

Table 2-21: **Metric Count Log Enable Register Bit Definitions (0x010C to 0x19C)**

Bits	Name	Access	Reset Value	Description
31-0	Counter Cut Off Value	Read / Write	0	Metric counter cut off value to enable event logging.
31-16	Range_high	Read / Write	0xFFFF	Higher count bound for enabling the event log
15-0	Range_low	Read / Write	0x0000	Lower count bound for enabling the event log

Notes:
1. Number of Metric Count Log Enable Registers that exist in the system is based on No of Event Counters parameter set in the GUI.
2. x varies from 0 to No of Event Counters-1.
3. Refer to the register space table for addresses of all Metric Count Log Enable Registers.

Sampled Metric Counter x Registers (SMCR)

The Metric Counters are captured into the Sampled Metric Counter registers when there is a capture event. The capture event can be an external event passed to the core or an overflow event of the sample interval counter. These registers are in AXI clock domain. The bit definition and accessibility of this register are shown in Table 2-22.

Table 2-22: **Sampled Metric Counter Register Bit Definitions (0x0200)**

Bits	Name	Access	Reset Value	Description
31-0	Sampled Metric Counter (Accumulator)	Read	0	Contains the sampled Metric Counter value

Notes:
1. Number of Sampled Metric Counters that exist in the system is based on No of Event Counters parameter set in the GUI.
2. x varies from 0 to No of Event Counters-1.
3. Refer to the register space table for addresses of all the sampled metric counters.

Sampled Incrementer x Registers (SIR)

The Incrementers are captured into the Sampled Incrementer registers when there is a capture event. The capture event can be an external event passed to the core or an overflow event of the sample interval counter. These registers are in AXI clock domain. The bit definition and accessibility of this register are shown in Table 2-23.

Table 2-23: **Sampled Incrementer Register Bit Definitions (0x0204)**

Bits	Name	Access	Reset Value	Description
31-0	Sampled Incrementer	Read	0	Contains the sampled Incrementer value

Notes:

1. Number of Sampled Metric Counters that exist in the system is based on No of Event Counters parameter set in the GUI.
2. x varies from 0 to No of Event Counters-1.
3. Refer to the register space table for addresses of all the sampled metric counters.

Control Register (CR)

The Control Register is a 32-bit register. This register is used to enable and reset the metrics counters inside the core. It is also used in enabling the event logging logic inside the core. The bit definition and accessibility of this register are shown in Table 2-24.

Table 2-24: **Control Register Bit Definitions (0x0300)**

Bits	Name	Access	Reset Value	Description
31-26	Reserved	N/A	N/A	Reserved
25	Streaming_FIFO_Reset	Read/Write	0	1: Resets the streaming FIFO
24	Reserved	N/A	N/A	Reserved
23-18	Reserved	N/A	N/A	Reserved
17	Global_Clk_Cnt_Reset	Read/Write	0	1: Resets the free-running Global Clock Counter
16	Global_Clk_Cnt_En	Read/Write	0	1: Enables the free-running Global Clock Counter
15-10	Reserved	N/A	N/A	Reserved
9	Use_Ext_Trig_Log	Read/Write	0	1: Uses the external trigger to start the event log
8	Enable_Event_Log	Read/Write	0	1: Enables event logging
7-3	Reserved	N/A	N/A	Reserved
2	Use_Ext_Trig	Read/Write	0	1: Uses the external trigger to start the metric counters
1	Metrics_Cnt_Reset	Read/Write	0	1: Resets all metric counters and sampled metric counters in the monitor
0	Metrics_Cnt_En	Read/Write	0	1: Enables all metric counters in the monitor

Latency ID Register (LIDR)

The write ID and Read ID that need to be monitored for the latency metric calculation has to be configured in the register. The bit definition and accessibility of this register are shown in Table 2-25.

Table 2-25: **Latency ID Register Bit Definitions (0x304)**

Bits	Name	Access	Reset Value	Description
31-16	Unused	-	-	N/A
15-8	RID	Read/Write	0x00	Read ID to capture read latency metrics
7-0	WID	Read/Write	0x00	Write ID to capture write latency metrics

Flag Enable Control Register (FECR)

The Flag Enable Control Register is a 32-bit register. This register is used to enable the events that are used to log the data into streaming FIFO. The bit definition and accessibility of this register are shown in Table 2-26.

Table 2-26: **Flag Enable Control Register Bit Definitions (0x0400)**

Bits	Name	Access	Reset Value	Description
31	Enable metric counter 9 Flag	Read/Write	0	1: Enables event logging on metric counter 9 value crossing the count given in cut off register 9
30	Enable Metric Counter 8 Flag	Read/Write	0	1: Enables event logging on metric counter 8 value crossing the count given in cut off register 8
29	Enable Metric Counter 7 Flag	Read/Write	0	1: Enables event logging on metric counter 7 value crossing the count given in cut off register 7
28	Enable Metric Counter 6 Flag	Read/Write	0	1: Enables event logging on metric counter 6 value crossing the count given in cut off register 6
27	Enable Metric Counter 5 Flag	Read/Write	0	1: Enables event logging on metric counter 5 value crossing the count given in cut off register 5
26	Enable Metric Counter 4 Flag	Read/Write	0	1: Enables event logging on metric counter 4 value crossing the count given in cut off register 4
25	Enable Metric Counter 3 Flag	Read/Write	0	1: Enables event logging on metric counter 3 value crossing the count given in cut off register 3
24	Enable Metric Counter 2 Flag	Read/Write	0	1: Enables event logging on metric counter 2 value crossing the count given in cut off register 2

Table 2-26: **Flag Enable Control Register Bit Definitions (0x0400) *(Cont'd)***

Bits	Name	Access	Reset Value	Description
23	Enable Metric Counter 1 Flag	Read/Write	0	1: Enables event logging on metric counter 1 value crossing the count given in cut off register 1
22	Enable Metric Counter 0 Flag	Read/Write	0	1: Enables event logging on metric counter 0 value crossing the count given in cut off register 0
21	Enable Sample counter Lapse Flag	Read/Write	0	1: Enables event logging on sample counter lapse
20	Enable Global Clock Count Overflow Flag	Read/Write	0	1: Enables event logging on global clock counter overflow
19	Enable External Event Start Flag	Read/Write	0	1: Enables event logging on Event start signal
18	Enable External Event Stop Flag	Read/Write	0	1: Enables event logging on Event stop signal
17	Enable External Event Flag	Read/Write	0	1: Enables event logging on Event signal
16	Enable Software-written data Flag	Read/Write	0	1: Enables event logging on write to Software-written data register
15-7	Reserved	N/A	N/A	Reserved
6	Enable Last Read Flag	Read/Write	0	1: Enables event logging on last read of a transaction
5	Enable First Read Flag	Read/Write	0	1: Enables event logging on first read of a transaction
4	Enable Read Addr Flag	Read/Write	0	1: Enables event logging on read address latch
3	Enable Response Flag	Read/Write	0	1: Enables event logging on response of a transaction
2	Enable Last Write Flag	Read/Write	0	1: Enables event logging on last write of a transaction
1	Enable First Write Flag	Read/Write	0	1: Enables event logging on first write of a transaction
0	Enable Write Addr Flag	Read/Write	0	1: Enables event logging on write address latch

Software-Written Data Register (SWDR)

The Software-Written Data Register is a 32-bit register. This register value is written into the streaming FIFO upon write to this register. The bit definition and accessibility of this register are shown in Table 2-27.

Table 2-27: **Software-Written Data Register Bit Definitions (0x0404)**

Bit	Name	Access	Reset Value	Description
31-0	Software-written Data	Read/Write	0x00	Software-Written Data

Designing with the Core

This chapter includes guidelines and additional information to make designing with the core easier.

General Design Guidelines

The following steps are recommended for all designs using the AXI Performance Monitor core for performance counting and event logging.

Performance Counting using Event Log Module

1. Instantiate the Performance Monitor core in the system. See Figure 3-1.
2. Configure the core slot protocol based on the agents connected to the monitor slots.
3. Verify `core_aclk` is the fastest clock in the design.
4. Program the configuration registers.
5. Write to Metric Selector Registers with the metric type and targeted agent.
6. Enable the metrics and global counter (Control Register). Use the external trigger pulse by enabling it in the Control Register to start the metric counts if required. External trigger usage is an optional method to enable the metric counters.
7. Read Metric Counters/Incrementers/Sampled Metric Counters/Sampled Incrementers/ Global Clock Counter based on the requirement.

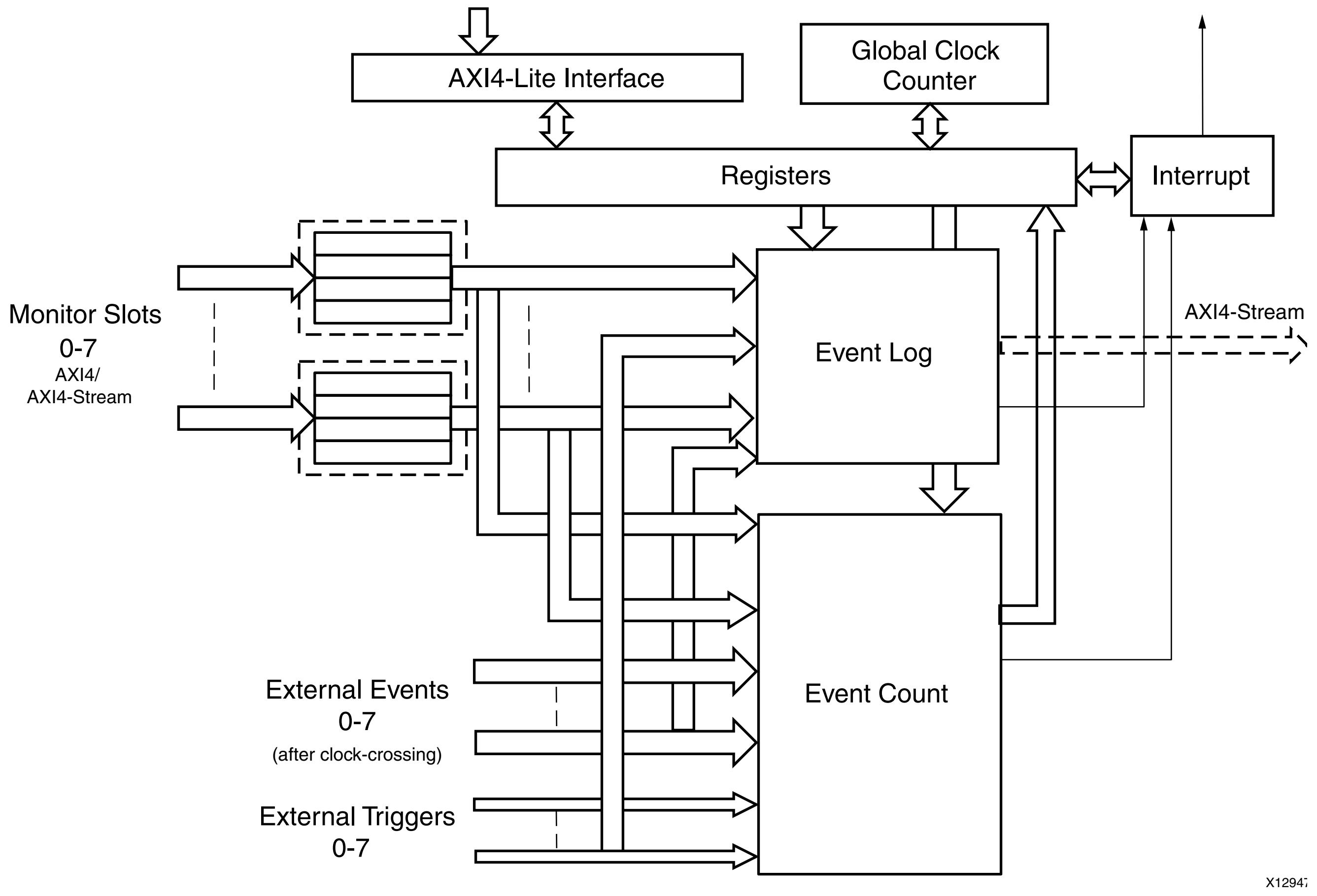


Figure 3-1: **Block Diagram of AXI Performance Monitor**

Event Logging using Event Count Module

1. Instantiate the Performance Monitor core in any system.
2. Configure the core based on the agents connected to the monitor slots.
3. Verify `core_aclk` is the fastest clock in the design.
4. Program the configuration registers.
5. Enable the Event Log bit in Control register.
6. Enable the required flags to capture events in the Event Log through the Flag Enable register.
7. Load the target metric counts in the Metric Count Log Enable register *x* and set the corresponding flag for cross probing. If the metric count reaches the cut off count given, then it automatically starts the event logging.
8. Read out the data from streaming interface of the core.

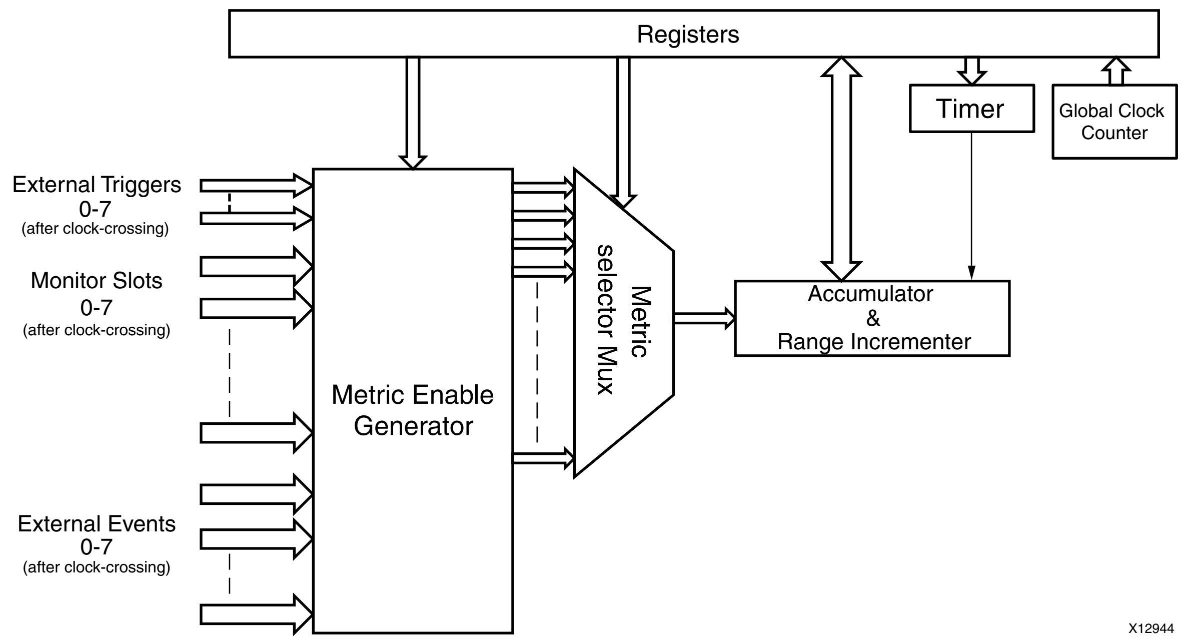

Registers
Timer
Global Clock Counter
External Triggers 0-7 (after clock-crossing)
Monitor Slots 0-7 (after clock-crossing)
External Events 0-7 (after clock-crossing)
Metric Enable Generator
Metric selector Mux
Accumulator & Range Incrementer
X12944

Figure 3-2: **Event Count Module**

Programming Sequence

This section describes how to program the AXI Performance Monitor for performance counting and event logging.

Capturing Metric Counts with Sample Interval Counter

This section uses a system built with eight monitor slots, eight external events, and ten counters. In this example system Slot 7 is connected to the streaming interface, and Slot 2 is connected to the memory map interface. In this use case, the steps to capture the metric counts with the sample interval counter are as follows:

1. Reset the global clock counter and metric counters by writing 0x0020002 into the Control register (0x300).
2. Configure the metric selection registers (0x44, 0x48, 0x4C) for the required slot metrics. The following list details the configuration that should be used with each metric selection register:
   - Write Metric Selection Register 0 (0x44) with 0xF6F1F2F0
     - Metric Counter 0: Slot 7 Transfer Cycle count (F0)
     - Metric Counter 1: Slot 7 Data Byte count (F2)
     - Metric Counter 2: Slot 7 Packet count (F1)

    - Metric Counter 3: Slot 7 Master Idle count (F6)
  - Write Metric Selection Register 1 (0x48) with 0x26222120
    - Metric Counter 4: Slot 2 Write Transaction count (20)
    - Metric Counter 5: Slot 2 Read Transaction count (21)
    - Metric Counter 6: Slot 2 Write Byte count (22)
    - Metric Counter 7: Slot 2 Total Write latency (26)
  - Write Metric Selection Register 2 (0x4C) with 0x00005E1E
    - Metric Counter 8: External Event 0 count (1E)
    - Metric Counter 9: External Event 2 count (5E)

3. Enable Global Interrupt (0x30) with 0x00000001.
4. Enable Sample Interval Counter Overflow Interrupt (0x34) with 0x00000002.
5. Write the load value based on requirements in the Sample Interval Register (0x24) with 0x00001000 (4096 clocks).
6. Set the Load bit in the Sample Interval Control Register (0x28), that is 0x00000002.
7. Enable the Sample Interval Counter. Write 0x00000001 in the Sample Interval Control Register (0x28).
8. Enable Metric Counters. If required, enable the global clock counters by writing 0x00100001 in control register(0x300).

The Performance Monitor configuration is complete at this point, and the relevant slot transactions are now active (Slot 7, Slot 1, External Event 0 and External Event 2 for this example).

9. Wait for the Sample Interval Counter Overflow Interrupt (0x2 in 0x38), and then disable the sampled metric counters by writing 0x0000000 in 0x28.
10. Read the sampled metric counters (0x200, 0x210 and so on up to 0x290) to capture the configured metrics for a known period (4096 core clocks in this example).

Metric Counts for DMA/Memory Transfers

Follow these steps to obtain the metric counts for DMA/Memory transfers:

1. Reset the global clock counter and metric counters by writing 0x0020002 into the Control Register (0x300).
2. Configure the metric selection registers (0x44, 0x48, 0x4C) for the required slot metrics.
3. Enable the metric counters by writing 0x00000001 in the Control Register(0x300).
4. Start DMA/Memory transfers.

5. After the transfers are complete, read the metric counters (0x100, 0x110 and so on up to 0x190) to get the configured metrics.

Software Application Profiling

For software application profiling, set the global clock counter width to either 32 or 64 based on your requirements. After the appropriate width configuration for the global clock counter has been set, follow these steps:

1. Reset the global clock counter by writing 0x0020000 into the Control Register (0x300).
2. Enable the global clock counter by writing 0x0010000 into the Control Register (0x300).
3. Run the application.
4. Read the 32 or 64-bit global clock counter value (0x4, 0x0) at the end of the application or at a specific event in the application to get the runtime of application.

Write and Read Latency Distribution of Monitor Slot for Five Ranges

Assuming the core is configured to have a single monitor slot (C_NUM_MONITOR_SLOTS =1) and 10 metric counters (C_NUM_OF_COUNTERS = 10), follow these steps to obtain the write and read latency distribution:

1. Reset the metric counters by writing 0x0000002 into the Control Register (0x300).

   Configure the metric selection registers (0x44, 0x48, 0x4C) for the read and write latency of slot 0. Configure the metric counters 0 to 4 for read latency and metric counters 5 to 9 for write latency.

   - Write Metric Selection Register 0 (0x44) with 0x05050505
   - Write Metric Selection Register 1 (0x48) with 0x06060605
   - Write Metric Selection Register 2 (0x4C) with 0x00000606
2. Configure the range registers with low and high values.
   - Range registers for read latencies:
     - Write Range Register 0 (0x108) with 0x140000 (0 to20)
     - Write Range Register 1 (0x118) with 0x280015 (21 to 40)
     - Write Range Register 2 (0x128) with 0x3C0029 (41 to 60)
     - Write Range Register 3 (0x138) with 0x50003D (61 to 80)
     - Write Range Register 4 (0x148) with 0x640051 (81 to 100)
   - Range registers for write latencies:

- Write Range Register 0 (0x158) with 0x140000 (0 to20)
- Write Range Register 1(0x168) with 0x280015 (21 to 40)
- Write Range Register 2 (0x178) with 0x3C0029 (41 to 60)
- Write Range Register 3 (0x188) with 0x50003D (61 to 80)
- Write Range Register 4 (0x198) with 0x640051 (81 to 100)

3. Enable Metric Counters by writing 0x00000001 in the Control Register (0x300).
4. Slot 0 transactions can be started to capture read and write latencies.
5. After the Slot 0 transactions, read the incrementers (0x104, 0x114 and on up to 0x194) for the read and write latencies distribution in the given ranges.

By following this use case to load and enable sample interval counters, it is possible to get the data in sampled incrementers (0x204, 0x214 and on up to 0x294) after the 0x0024 clocks which are loaded in the Sample Interval register.

Event Log on First Write, First Read, Software-Written Data and External Event Flags

This use case assumes that event logging is enabled and the other event log parameters (FIFO Width, Enable TID, Enable TDEST, Enable TUSER, Enable ID, and Enable Length) are set properly. The steps to capture the event log on first write, first read, software-written data and external event flags is:

1. Reset the Streaming FIFO by writing 0x2000000 into the Control Register (0x300).
2. Enable the flags that are required to capture events in the event log by writing to the Flag Enable Register (0x400). For example, to enable First write, First Read, External Event and Software-Written Data flags, write 0x30022 in to the Flag Enable Register.
3. Enable the external event log bit in the Control Register (write 0x100 in 0x300).
4. Write to the Software-Written Data Register with the data required to log.
5. Offload the log data through the Streaming Interface.

Metric Counts for DMA/Memory Transfers with External Trigger Pulse

Follow these steps to obtain the metric counts for DMA/Memory transfers:

1. Reset the global clock counter and metric counters by writing 0x0020002 into the Control Register (0x300).
2. Configure the metric selection registers (0x44, 0x48, 0x4C) for the required slot metrics.

3. Enable the metric counters and set them using an external trigger by writing 0x00000005 in the Control Register (0x300).
4. Apply the external trigger start pulse and then start DMA/Memory transfers to capture in metric counters.
5. After the transfers are complete, read the metric counters (0x100, 0x110 and so on up to 0x190) to get the configured metrics.
6. Apply the reset and follow the same steps to enable the external trigger pulse again to start the metric counters.

Cross Probing

Cross probing is capturing the Event Log on First Write, First Read and External Event Flags when the interested metric counter 2 exceeds the expectation set. This use case assumes the event logging and event counters are enabled, and that the other event log parameters are set properly.

The steps to capture the event log on first write, first read and external event flags is:

1. Reset the Streaming FIFO and metric counters by writing 0x2000002 into the Control Register (0x300).
2. Load the Metric Count Event Log Enable register 2 (0x12C) with the expected value. This starts the event log once metric counter 2 reaches the expected value.
3. Enable the flags that are required to capture events in the event log by writing to the Flag Enable Register (0x400). For example, do this to enable first write, first read, external event and the metric counter 2 flag to enable cross probing. Write 0x1020022 into the Flag Enable Register.
4. Configure the metric selection registers (0x44, 0x48, 0x4C) with the interested metrics.
5. Set the enable metric count enable bit in the Control Register (write 0x001 in 0x300).
6. Start the transactions of interest over the connected monitor slot.
7. Offload the log data through the Streaming Interface.

Clocking

The AXI Performance Monitor supports asynchronous clock domains for core clock and other available clock domains. The synchronization for each clock domain must be enabled if required. Make the proper clock connection to all the clocks on the available interfaces. The AXI Performance Monitor has the following clock domains :

- AXI4-Lite clock domain is clocked by `s_axi_aclk`.

- Core clock domain is clocked by `core_aclk`.
- AXI4 Slot clock domain is clocked by `slot_n_axi_aclk`.
- AXI4-Stream clock domain is clocked by `slot_n_axis_aclk`.
- External event clock domain is clocked by `ext_clk_n`.
- Event log clock domain is clocked by `m_axis_aclk`.

Resets

There is an active-Low reset signal for each available clock domain. The only exception is `reset_event` which is an active-High signal used for the sampled metric counters. The reset signal must be synchronous to the corresponding clock signal.

Customizing and Generating the Core

This chapter includes information about using Xilinx tools to customize and generate the core in the Vivado™ Design Suite environment.

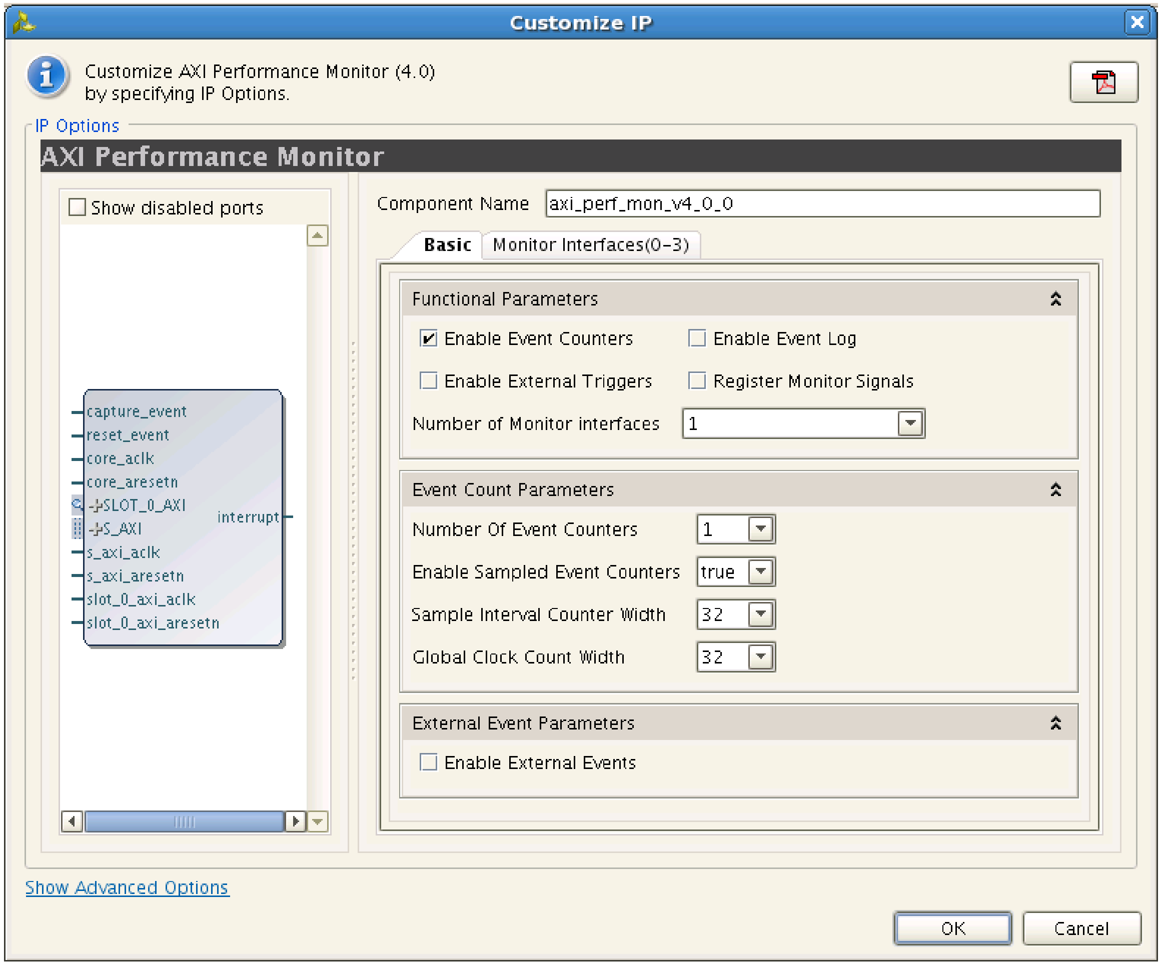
GUI

The AXI Performance Monitor can be found in **Embedded_Processing\Debug & Verification** in the Vivado IP Catalog.

To access the AXI Performance Monitor, do the following:

1. Open an existing project or create a new project in Vivado.
2. Open the IP catalog and navigate to any of the above taxonomies.

Double-click on AXI Performance Monitor to bring up the AXI Performance Monitor GUI.


Customize IP

Customize AXI Performance Monitor (4.0)
by specifying IP Options.

IP Options

AXI Performance Monitor

☐ Show disabled ports

Component Name: axi_perf_mon_v4_0_0

Basic | Monitor Interfaces(0-3)

Functional Parameters

☑ Enable Event Counters ☐ Enable Event Log

☐ Enable External Triggers ☐ Register Monitor Signals

Number of Monitor interfaces: 1

Event Count Parameters

Number Of Event Counters: 1

Enable Sampled Event Counters: true

Sample Interval Counter Width: 32

Global Clock Count Width: 32

External Event Parameters

☐ Enable External Events

Show Advanced Options

Figure 4-1: **Vivado Basic Screen**

The GUI has been split into tabs for ease-of-use. The Basic tab contains most of the configuration options. Each monitor slot has a separate tab for configuring the slot-related parameters.

Component Name

The component name is the base name of the output files generated for the core. Names must begin with a letter and can be composed of any of the following characters: a to z, 0 to 9, and "_".

Basic Options

This section describes the GUI options for the AXI Performance Monitor core.

Functional Parameters

- **Enable Event Counters**: Enables the event count logic inside the core.

- **Enable Event Log**: Enables the event log logic inside the core.
- **Enable External Triggers**: Enables external triggers to start and stop event counters and event log.
- **Register Monitor Signals**: Registers all monitor signals at the interface and as a result improves timing.
- **Number of Monitor Interfaces**: Indicates the number of monitor slots.

Event Count Parameters

- **Number of Event Counters**: Indicates the number of metric counters.
- **Enable Sampled Event Counters**: True enables the sampled metric counters inside the core.
- **Sample Interval Counter Width**: Indicates the width of Sample Interval Counter. The value can be 32 or 64.
- **Global Clock Counter Width**: Indicates the width of the Global Clock Counter. The value can be 32 or 64.

External Event Parameters

- **Enable External Events**: Enables external events to monitor and log.
- **Enable Synchronizers for Event (0 ... 7)**: Enables FIFO for synchronizing external event 0 signals to the core clock domain

Monitor Interfaces

Slot (0 to 7)

- **AXI Protocol**: Indicates if the connected agent to the monitor slot is AXI3 or AXI4 or AXI4-Stream.
- **Enable Synchronizer**: Enables FIFO for synchronizing AXI monitor slot signals to the core clock domain.
- **ID Width**: Indicates the ID width of the AXI4 slot.
- **Data Width**: Indicates the DATA width of the AXI4 slot.
- **Address Width**: Indicates the Address width of the AXI4 slot.
- **TUSER Width**: Indicates the ID width of the AXI4-Stream slot.
- **TDATA Width**: Indicates the Data width of the AXI4-Stream slot.
- **TDEST Width**: Indicates the TDEST width of the AXI4-Stream slot.
- **TID Width**: Indicates the TID width of the AXI4-Stream slot.

Log Parameters

Event Log Streaming Parameters

- **Use Synchronous FIFO**: Enables the Synchronous FIFO. By default the core uses Asynchronous FIFO.
- **Depth**: Event Log Streaming FIFO depth
- **TID Width**: Event Log Streaming TID Width

AXI4-Stream Monitor Slot Parameters

- **Enable Ids in Log Data**: Captures the AXI4 Slot IDs in the Event Log Data.
- **Enable Transaction Length in Log Data**: Captures the AXI4 slot transaction lengths in Event Log Data.
- **Enable TDEST in Log Data**: Captures the AXI4-Stream slot transaction TDEST in the Event Log Data.
- **Enable TUSER in Log Data**: Captures the AXI4-Stream slot transaction TUSER in the Event Log Data.
- **Enable TID in Log Data**: Captures the AXI4-Stream slot transaction TID in the Event Log Data.

Constraining the Core

The following XDC constraints are added in core-level XDC file.

```
set clk_domain_a [get_clocks -of_objects [get_ports s_axi_aclk]]
set clk_domain_b [get_clocks -of_objects [get_ports core_aclk]]
set_false_path -from [all_registers -clock $clk_domain_a] -to [all_registers -clock
$clk_domain_b]
set_false_path -from [all_registers -clock $clk_domain_b] -to [all_registers -clock
$clk_domain_a]
```

Debugging

This appendix provides information for using the resources available on the Xilinx Support website, debug tools, and other step-by-step processes for debugging designs that use the AXI Performance Monitor.

The following topics are included in this appendix:

- Finding Help on Xilinx.com
- Debug Tools
- Hardware Debug
- Interface Debug

Finding Help on Xilinx.com

To help in the design and debug process when using the AXI Performance Monitor, the Xilinx Support web page (www.xilinx.com/support) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for opening a Technical Support Web Case.

Documentation

This product guide is the main document associated with the AXI Performance Monitor. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page (www.xilinx.com/support) or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page (www.xilinx.com/download). For more information about this tool and the features available, open the online help after installation.

Technical Support

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled DO NOT MODIFY.

Xilinx provides premier technical support for customers encountering issues that require additional assistance.

To contact Xilinx Technical Support:

1. Navigate to www.xilinx.com/support.
2. Open a WebCase by selecting the WebCase link located under Support Quick Links.

When opening a WebCase, include:

- Target FPGA including package and speed grade.
- All applicable Xilinx Design Tools and simulator software versions.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which file(s) to include with the WebCase.

Master Answer Record for AXI Performance Monitor

AR 5442

Debug Tools

There are many tools available to address AXI Performance Monitor design issues. It is important to know which tools are useful for debugging various situations.

Vivado Lab Tools

Vivado inserts logic analyzer and virtual I/O cores directly into your design. Vivado Lab Tools allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature represents the functionality in the Vivado IDE that is used for logic debugging and validation of a design running in Xilinx FPGA devices in hardware.

The Vivado logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues.

Many of these common issues can also be applied to debugging design simulations. Details are provided on:

- General Checks
- Core-Specific Checks

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the LOCKED port.

Core-Specific Checks

Perform the following checks to further the debugging process.

- Check that the metric counters and event log are enabled through parameter configuration.
- Check the software configuration as described in Programming Sequence in Chapter 3.
- If Metric Counts and Event Log are coming as zero, check whether the monitor slot interfaces are active.

Interface Debug

AXI4-Lite Interfaces

Write a known value to any Read/Write register and read back the value from the register. If data does not match, ensure that the following conditions are met:

- The `s_axi_aclk` is connected and toggling.

- The interface is not being held in reset, and `s_axi_aresetn` is an active-Low reset.
- The main core clocks are toggling.
- If the simulation has been run, verify in simulation that the waveform is correct for accessing the AXI4-Lite interface.

AXI4-Stream Interfaces

If data is not being transmitted or received, check the following conditions:

- If transmit `m_axis_tready` is stuck low following the `m_axis_tvalid` input being asserted, the core cannot send data.
- If the receive `m_axis_tvalid` is stuck Low, the core is not sending data.
- Check that the `m_axis_aclk` input is connected and toggling.
- Check core configuration for event log data as described in Programming Sequence in Chapter 3.

Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:

www.xilinx.com/company/terms.htm.

References

These documents provide supplemental material useful with this user guide:

1. *ARM AMBA AXI4 Protocol Version: 2.0 Specification*
2. *AMBA AXI4-Stream Protocol Version: 1.0 Specification*
3. *7 Series FPGAs Overview* (DS180)
4. *Vivado Design Suite User Guide: Designing with IP* (UG896)
5. *Xilinx AXI Reference Guide* (UG761)
6. Vivado Design Suite Documentation

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/24/2012	1.0	Initial Xilinx release.
05/22/2012	1.1	Added Zynq-7000 support.
07/25/2012	2.0	Added support for Vivado Design Suite. Added Event Log module and changed Event Count logic to be symmetric.
10/16/2012	3.0	• Added External Event ports 2-7. • Added support for capturing any slot any metric at a time. • Event Log data is now captured based on the configuration to reduce the width of log data.
12/18/2012	4.0	• Updated core to v3.00a, Vivado Design Suite to v2012.4 and Xilinx Platform Studio (XPS) to v14.4. • Added the External Trigger Interface. • Added Metric Count Log Enable Registers. • Added Appendix G, Debugging.
03/20/2013	5.0	• Updated core to v4.0. • Removed support for Xilinx Platform Studio (XPS). • Added the Latency ID Register (LIDR).
06/19/2013	4.0	• Revision number advanced to 4.0 to align with core version number. • Updated description for the Metrics_Cnt_Reset bit of the Control Register in Table 2-24.

Notice of Disclaimer