AXI Performance Monitor v5.0

LogiCORE IP Product Guide

Vivado Design Suite

PG037 October 5, 2016

XILINX
ALL PROGRAMMABLE™

Table of Contents

Chapter 6: Test Bench

Appendix A: Migrating and Upgrading

Appendix B: Debugging

Appendix C: Additional Resources and Legal Notices

Introduction

The LogiCORE™ IP AXI Performance Monitor core enables AXI system performance measurement for multiple slots (AXI4/AXI3/AXI4-Stream/AXI4-Lite). This core captures configurable real-time performance metrics for throughput and latency for connected AXI interfaces. In addition, the AXI Performance Monitor core 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/AXI3/AXI4-Stream/AXI4-Lite) 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)	UltraScale+™ Families, UltraScale™ Architecture, Zynq®-7000, 7 Series
Supported User Interfaces	AXI4-Stream, AXI4-Lite, AXI3 and AXI4
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Verilog
Example Design	Verilog
Test Bench	Verilog
Constraints File	XDC
Simulation Model	None
Supported S/W Driver(2)	Standalone, Linux
Tested Design Flows(3)	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

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

Overview

The LogiCORE™ IP AXI Performance Monitor core measures major performance metrics for the AMBA AXI system. The Performance Monitor measures bus latency of a specific master/slave (AXI4/AXI3/AXI4-Stream/AXI4-Lite) 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 core (APM) supports three modes for analyzing system behavior on AXI interfaces:

- **Advanced**: Advanced mode supports two major functions on AXI4/AXI3/AXI4-Stream/AXI4-Lite interfaces.
  - Event Logging: The APM logs the specified AXI monitor slots' events (User Configuration) 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.
- **Profile**: Profile mode provides the event counting functionality of the APM with less user configuration to analyze system behavior on AXI4/AXI3/AXI4-Lite interfaces. It provides event counting for fixed metrics on each slot. The number of counters per slot and the metric for each counter are pre-defined. Profile mode does not support the AXI4-Stream interface and external event metrics.
- **Trace**: Trace mode provides event logging functionality of the APM with less user configuration than Advanced mode. All the flags are enabled/disabled through the Vivado® Integrated Design Environment (IDE) options when generating the core and cannot be modified after core generation. Trace mode does not support the AXI4-Stream interface.

Advanced Mode

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

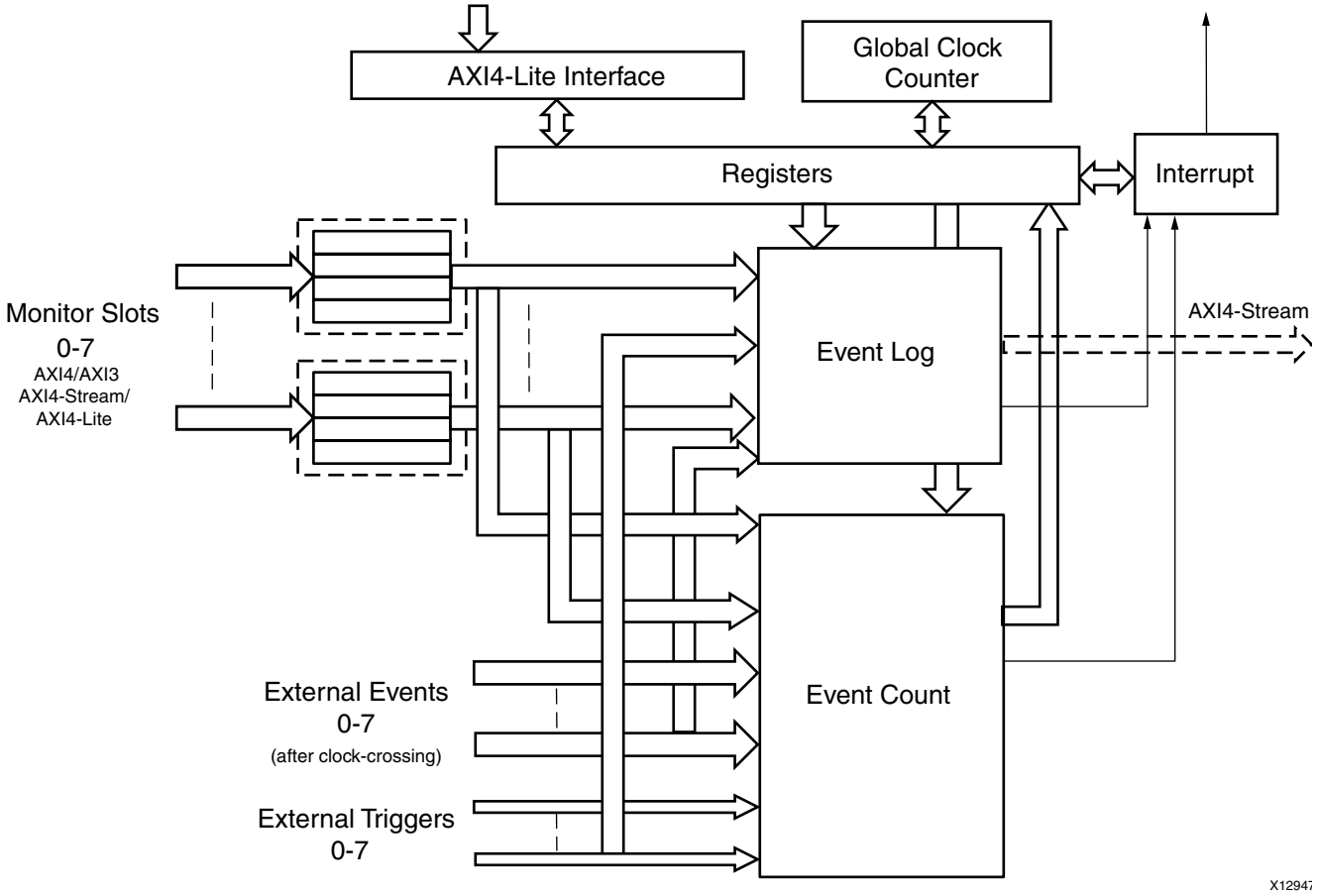

Figure 1-1: **Block Diagram of AXI Performance Monitor core**

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 following flags can be selected for AXI4/AXI3/AXI4-Lite event logging:

- 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/AXI3/AXI4-Lite includes:

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

Note: For AXI4-Lite interfaces, the ID width is 0, and AWLEN/ARLEN is set to 0.

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 your selection for enabling ID and Transaction Length for Log data when configuring the core. AXI4-Stream log data width is based on your 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	Always	Packet type. 1= SW data register
Time Stamp Difference	16	Always	Contains the time stamp difference between the previous event (write to FIFO) and the current event.
Loop Event	1	Always	Is '1' when the time stamp difference exceeds 2^16.
SW Data Register	32	Always	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	Always	Packet type. 0=monitor slots log
Time Stamp Difference	16	Always	Contains the time stamp difference between the previous event (write to FIFO) and the current event.
Loop Event	1	Always	Is '1' when the time stamp difference exceeds 2^16.
External Event 0 Flags	3	Always	External Event 0 start, stop and Event Flags are captured.
Slot 0 Flags	7	Slot 0 AXI Protocol = AXI4/AXI3/ AXI4-Lite	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(1) x 4) + (Enable Length x 16)		AWID, BID, ARID, RID, AWLEN, ARLEN.

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

Name	Width	Is Valid	Description
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.
Slot 1 Flags	7	(Slot 1 AXI Protocol = AXI4/AXI3/ AXI4-Lite) && (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.

Notes:

1. The core sees the ID width as 1 for IDE selections of 1 or 0.

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.

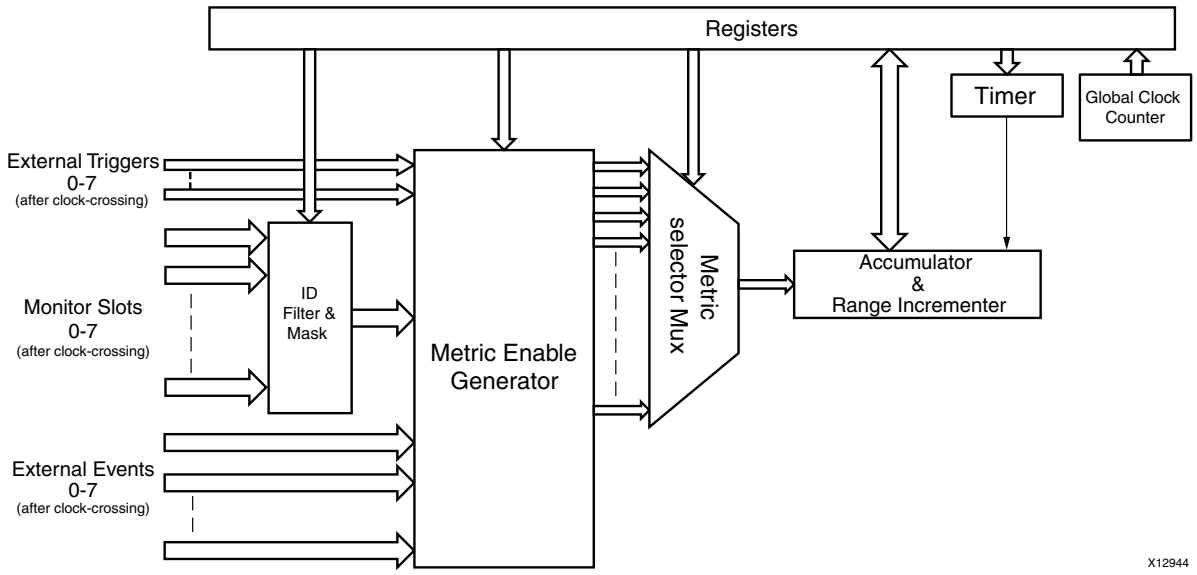

Figure 1-2: **Event Count Module**

Event Counting consists of a configurable number of Accumulator and Range Incrementer blocks up to 10. (for example, metric counters). ID filtering/masking is applied on all the events based on the configured ID and mask bits in the configuration registers. ID filtering is optional and applies based on the filtering enable bit in the control register. 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/AXI3/AXI4-Stream/AXI4-Lite 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. 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.

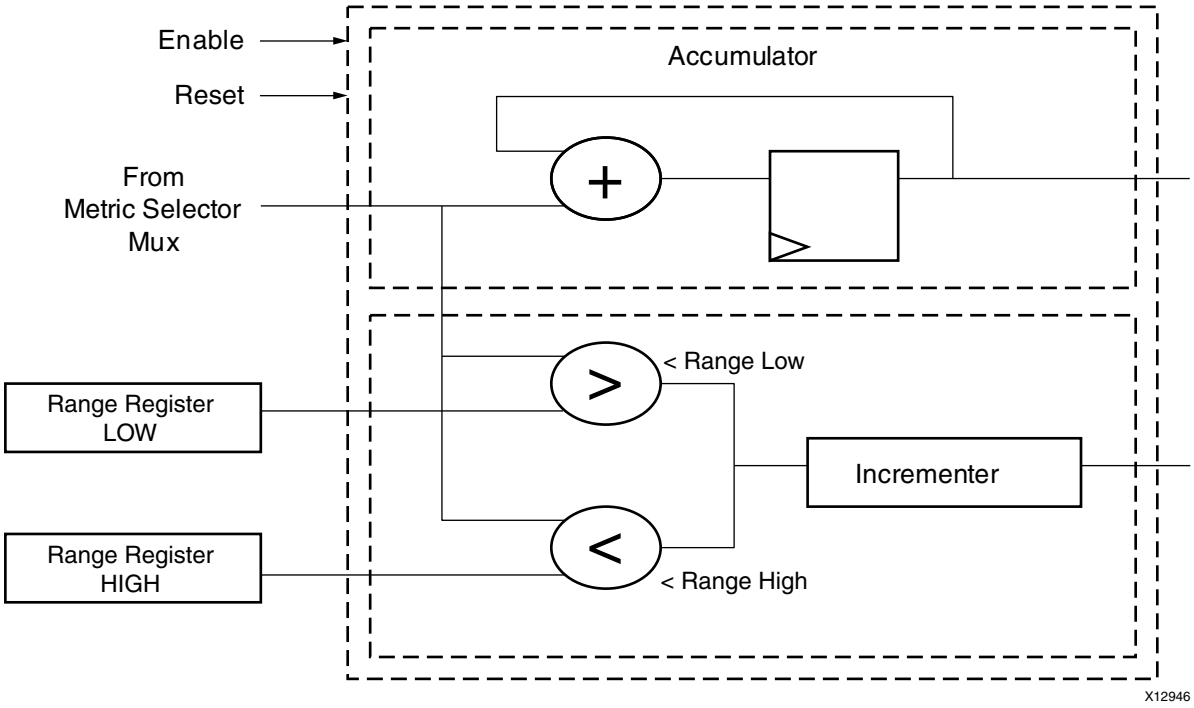

Figure 1-3: **Accumulator and Range Incrementer**

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 following sampling methods capture the accumulator values and log them in the corresponding sampled metric counters:

- **External capture event**: When the external capture event is set, the metric counter values are loaded into the corresponding sample metric counters. In addition to the capture event, there is a reset event that resets the metric counters and sample metric counters. There is no register configuration for this method of capturing.
- **Sample interval timer**: The sample interval timer is set with the number of clocks to count down. After the timer expires, the metric counter values are loaded into sampled metric counters. The metric counters are automatically reset after loading. There is a bit in sample interval control register to disable the auto-reset of metric counters after capturing.
- **Sample register read**: When there is a read to the sample register, all the metric counters are loaded into the sample metric counters. The data corresponding to the sample register read is a free running count value. This method of sampling is useful when external memory mapped master samples the data. The counter value returned by this register can be used by the external master to determine the timestamp for the earlier sample. The metric counters are automatically reset after loading. There is a bit

in the sample interval control register to disable the auto-reset of the metric counters after capturing.

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/AXI3/AXI4-Lite agent:

- **Write Transaction Count**: Total number of write requests by/to the agent.
- **Read Transaction Count**: Total number of read requests given by/to the agent.
- **Read Latency**: The time from the start of read address issuance/acceptance to the beginning/ending of the read data transaction. Latency start and end point selection is user configurable. By default the read latency is from the read address issuance by the master interface to the last read.
- **Write Latency**: The time from the start of the write address issuance/acceptance to the first/last write to the slave. Latency start and end point selection is configurable. By default, the write latency is from the write address issuance to the last write.
- **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.

Profile Mode

Event counting in profile mode is shown in Figure 1-4.

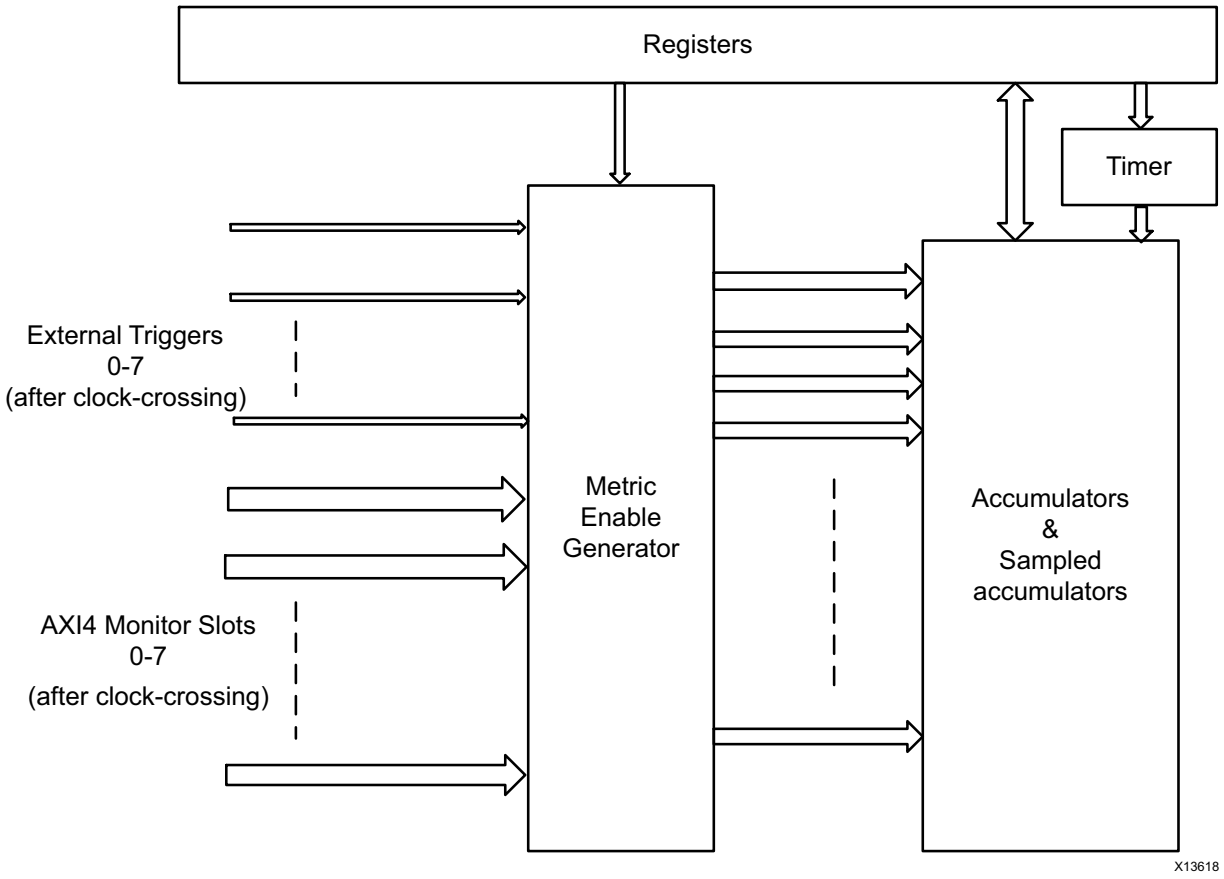

Figure 1-4: **Event Counting in Profile Mode**

Event counting in profile mode consists of a fixed number of accumulators for each AXI4/AXI3/AXI4-Lite slot. All the events that can be counted are detected and given to the accumulator which calculates the aggregate value. There is no selection of events, and in this mode, event counting is done only on AXI4/AXI3/AXI4-Lite monitor slots.

The metric counters in profile mode 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 slot. The external trigger pulse also needs the enable external trigger bit to be set in the control register. The external triggers control the start and stop of counters.

The following sampling methods are provided to capture the accumulator values and log them in the corresponding sampled metric counters. There are no incrementers or range registers in profile mode.

- **External capture event**: When the external capture event is set, the metric counter values are loaded into the corresponding sample metric counters. In addition to the capture event, there is a reset event that resets the metric counters and sample metric counters. There is no register configuration for this method of capturing.
- **Sample interval timer**: The sample interval timer is set with the number of clocks to count down. After the timer expires, the metric counter values are loaded into sampled metric counters. The metric counters are automatically reset after loading. There is a bit in the sample interval control register to disable the auto-reset of metric counters after capturing.
- **Sample register read**: When there is a read to the sample register, all the metric counters are loaded into the sample metric counters. The data corresponding to the sample register read is a free running count value that starts with first sample register read. The metric counters are automatically reset after loading. There is a bit in the sample interval control register to disable the auto-reset of metric counters after capturing.

Metrics computed for an AXI4/AXI3/AXI4-Lite agent in profile mode:

- **Write Transaction Count**: Total number of write requests by/to the agent.
- **Write Byte Count**: Total number of bytes written by/to the agent. This metric is helpful when calculating the throughput of the system.
- **Write Latency**: The time from the start of the write address issue/acceptance to the last/first write. The write latency start and end points are configurable through the control register. By default, the latency calculated is between the write address issuance and the last write.
- **Read Transaction Count**: Total number of read requests given by/to the agent.
- **Read Byte Count**: Total number of bytes read from/by the agent.
- **Read Latency**: The time from the start of read address issuance/acceptance to the last/first read data service. The read latency start and end points are configurable through the control register. By default, the read latency is from the read address issuance to the last read.

Note: ID filtering is not supported in profile mode.

Trace Mode

In trace mode, the APM provides event logging in a reduced dynamic configuration. It captures the specified AXI events, external events and the time stamp difference between two successive events into the streaming FIFO. The selection of events to be captured is set through parameter configuration. Streaming agents are not supported in trace mode. The flags that can be selected for AXI4/AXI3/AXI4-Lite agent are as follows:

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

In addition to the AXI4/AXI3/AXI4-Lite flags, there are SW data register write flags and external events flags. The event logging can also track the AXI4 data for analyzing the transactions. The log data for AXI4/AXI3/AXI4-Lite includes:

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

Note: For AXI4-Lite interfaces, the ID width is 0, and AWLEN/ARLEN is set to 0.

The packet format for the streaming FIFO is listed in Table 1-3 for two monitor slots. The width of the packet increases with the number of slots. The AXI4/AXI3/AXI4-Lite log data width is based on the configuration for enabling ID and Transaction Length for Log data in the Vivado IDE.

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 while configuring the core 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 1-3: **Monitor Log Packet in Trace Mode**

Name	Width	Is Valid	Description
Log ID	1	Always	Packet type: 0=monitor slots log
Time Stamp Difference	16	Always	Contains the time stamp difference between the previous event (write to FIFO) and the current event
Loop Event	1	Always	1 = the time stamp difference exceeds 2^{16}.
External Event 0 Flags	3	Always	External Event 0 start, stop and event flags are captured.
Slot 0 Flags	7	Always	Event Flags for Slot 0 (Last Read Flag, First Read Flag, Read Address Latch Flag, Response 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)	Always	AWID, BID, ARID, RID, AWLEN, ARLEN
External Event 1 Flags	3	Number of Monitor Slots > 1	External Event 1 start, stop and event flags are captured.
Slot 1 Flags	7	Number of Monitor Slots > 1	Event Flags for Slot 1 (Last Read Flag, First Read Flag, Read Address Latch Flag, Response 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)	Number of Monitor Slots > 1	AWID, BID, ARID, RID, AWLEN, ARLEN

Event logging can be enabled through the Control register; there are optional external triggers to start and stop the logging of each slot. Trace mode does not support cross probing.

TIP: *A single AXI transaction is expected to complete within 2^{32} clocks.*

Trigger feature

You can use this feature to debug the AXI transactions. When you enable the trigger parameter (C_EN_TRIGGER), the monitor log packet will send out the formats shown in Table 1-4 using a streaming interface. This feature is useful to debug the transactions in hung scenarios.

Table 1-4: **Monitor Log Packet in Trace Mode when address_flags, trigger logic parameters are enabled**

Name	Width	Is Valid	Description
Log ID	1	Always	Packet Type: 0=monitor slots log
Time Stamp Difference	16	Always	Contains the time stamp difference between the previous event (write to FIFO) and the current event
Loop Event	1	Always	1 = the time stamp difference exceeds 2^{16}.
Trigger Assertion	1	Always	1=trigger edge assertion packet
Slot 0 log Data	(Address width x 2) + (Enable Length x 16) + 8	Always	WADDR,AWLEN,WSP,WEP, WSPM,WEPM, RADDR, ARLEN, RSP, REP, RSPM, REPM
Slot 1 log Data	(Address width x 2) + (Enable Length x 16) + 8	Number of Monitor Slots > 1	WADDR,AWLEN,WSP,WEP, WSPM,WEPM,RADDR,ARLEN,RSP,REP,RSPM,REPM
Slot 2 log Data	(Address width x 2) + (Enable Length x 16) + 8	Number of Monitor Slots > 2	WADDR,AWLEN,WSP,WEP, WSPM,WEPM,RADDR,ARLEN,RSP,REP,RSPM,REPM
Slot 3 log Data	(Address width x 2) + (Enable Length x 16) + 8	Number of Monitor Slots > 3	WADDR,AWLEN,WSP,WEP, WSPM,WEPM,RADDR,ARLEN,RSP,REP,RSPM,REPM

Notes:

1. Abbreviations for the terms used in the table:

   WSP – Write Start Point, WSPM – Write Start Point Marker, WEP – Write Endpoint, WEPM – Write Endpoint Marker,

   WADDR –Write Address, RADDR – Read Address,

   RSP – Read Start Point, RSPM – Read Start Point Marker, REP – Read End Point, REPM – Read End Point Marker

2. As trigger can be driven externally at any time, the start point marker and end point marker assertion positions are not guaranteed in asynchronous mode i.e. when the core clock and slot clocks are different.

Applications

The AXI Performance Monitor core 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. For more information, see *Xilinx Software Development Kit User Guide: System Performance Analysis* [Ref 11]
- 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 core has the following known limitations:

- Bus contention metrics are not computed.
- Due to logic constraints, the advanced mode 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.
- Does not support interleaved transactions.
- Does not measure latencies of the transactions in which the gap between start and stop event exceeds 2^{32} clocks.

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) [Ref 4].

Maximum Frequencies

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

Note: Maximum frequency numbers for UltraScale™ architecture and 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-Stream	AXI4-Lite
Virtex-7	-1	200	200	180
Kintex-7		200	200	180
Artix-7		150	150	120
Virtex-7	-2	240	240	200
Kintex-7		240	240	200
Artix-7		180	180	140
Virtex-7	-3	280	280	220
Kintex-7		280	280	220
Artix-7		200	200	160

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 15 core clocks.

Resource Utilization

For details about resource utilization, visit Performance and Resource Utilization.

Port Descriptions

Table 2-2 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 2-2: **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.
interrupt	System	O	0x0	System Interrupt Output.
trigger_in	System	I		Trigger Input port
trigger_in_ack	System	O	0x0	Trigger ack port. Assertion of this bit by core indicates that the AXI Performance monitor core receives the trigger input and the corresponding logic is enabled.
AXI4-Lite Slave Interface				
s_axi_aclk	System	I		AXI Clock.
s_axi_aresetn	System	I		AXI Reset. Active-Low.

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

Signal Name	Interface	I/O	Initial State	Description
s_axi_*	S_AXI	-	-	See Appendix A of the *Vivado AXI Reference Guide* (UG1037)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/ SLOT_n_AXI4 LITE	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_n_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.
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.

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

Signal Name	Interface	I/O	Initial State	Description
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 2-3 shows all the AXI Performance Monitor registers and addresses in advanced mode. Table 2-4 shows the registers and addresses in profile and trace modes.

Note: The AXI4-Lite write access register is updated by the 32-bit AXI Write Data (`*_wdata`) signal, and is not impacted by the AXI Write Data Strobe (`*_wstrb`) signal. For write access, both the AXI Write Address Valid (`*_awvalid`) and AXI Write Data Valid (`*_wvalid`) signals should be asserted together.

Table 2-3: **Core Registers in Advanced Mode**

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 2-3: **Core Registers in Advanced Mode *(Cont'd)***

Address Offset	Register Name	Description
Sample Interval Registers		
0x0024	Sample Interval	Sample Interval Time Configuration Register
0x0028	Sample Interval Control Register	Sample Interval Control Register
0x002C	Sample Register	Sample Register: Reading this register initiates sampling of Metric Counters data to the Sample Metric Counters.
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

Table 2-3: **Core Registers in Advanced Mode *(Cont'd)***

Address Offset	Register Name	Description
0x0140	Metric Counter 4	Metric Counter 4 Register
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

Table 2-3: **Core Registers in Advanced Mode *(Cont'd)***

Address Offset	Register Name	Description
0x0230	Sampled Metric Counter 3	Sampled Metric Counter 3 Register
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	ID Register	ID Register
0x0308	ID Mask Register	Mask Register
Event Log Registers		
0x0400	Flag Enable Register	Flag Enable Register
0x0404	Software-written Data Register	Software-written Data Register

Table 2-4: **Core Registers in Profile and Trace Modes**

Address Offset	Register Name	Description
Sample Interval Registers		
0x0024	Sample Interval	Sample Interval Time Configuration
0x0028	Sample Interval Control Register	Sample Interval Control Register
0x002C	Sample Register	Sample Register
Interrupt Registers		
0x0030	Global Interrupt Enable Register	Global Interrupt Enable Register

Table 2-4: **Core Registers in Profile and Trace Modes *(Cont'd)***

Address Offset	Register Name	Description
0x0034	Interrupt Enable Register	Interrupt Enable Register
0x0038	Interrupt Status Register	Interrupt Status Register
Control Register		
0x0300	Control Register	Control Register
Metric Counters (Accumulators, Sampled Accumulators): Available in Profile Mode Only		
0x0100	Metric Counter 0(1)	Metric Counter for Slot 0 write byte count
0x0110	Metric Counter 1(1)	Metric Counter for Slot 0 write transaction count
0x0120	Metric Counter 2(1)	Metric Counter for Slot 0 write latency count
0x0130	Metric Counter 3(1)	Metric Counter for Slot 0 read byte count
0x0140	Metric Counter 4(1)	Metric Counter for Slot 0 read transaction count
0x0150	Metric Counter 5(1)	Metric Counter for Slot 0 read latency count
0x0160	Metric Counter 6(1)	Metric Counter for Slot 1 write byte count
0x0170	Metric Counter 7(1)	Metric Counter for Slot 1 write transaction count
0x0180	Metric Counter 8(1)	Metric Counter for Slot 1 write latency count
0x0190	Metric Counter 9(1)	Metric Counter for Slot 1 read byte count
0x01A0	Metric Counter 10(1)	Metric Counter for Slot 1 read transaction count
0x01B0	Metric Counter 11(1)	Metric Counter for Slot 1 read latency count
0x0500	Metric Counter 12(1)	Metric Counter for Slot 2 write byte count
0x0510	Metric Counter 13(1)	Metric Counter for Slot 2 write transaction count
0x0520	Metric Counter 14(1)	Metric Counter for Slot 2 write latency count
0x0530	Metric Counter 15(1)	Metric Counter for Slot 2 read byte count
0x0540	Metric Counter 16(1)	Metric Counter for Slot 2 read transaction count
0x0550	Metric Counter 17(1)	Metric Counter for Slot 2 read latency count
0x0560	Metric Counter 18(1)	Metric Counter for Slot 3 write byte count
0x0570	Metric Counter 19(1)	Metric Counter for Slot 3 write transaction count
0x0580	Metric Counter 20(1)	Metric Counter for Slot 3 write latency count

Table 2-4: **Core Registers in Profile and Trace Modes *(Cont'd)***

Address Offset	Register Name	Description
0x0590	Metric Counter 21(1)	Metric Counter for Slot 3 read byte count
0x05A0	Metric Counter 22(1)	Metric Counter for Slot 3 read transaction count
0x05B0	Metric Counter 23(1)	Metric Counter for Slot 3 read latency count
0x0700	Metric Counter 24(1)	Metric Counter for Slot 4 write byte count
0x0710	Metric Counter 25(1)	Metric Counter for Slot 4 write transaction count
0x0720	Metric Counter 26(1)	Metric Counter for Slot 4 write latency count
0x0730	Metric Counter 27(1)	Metric Counter for Slot 4 read byte count
0x0740	Metric Counter 28(1)	Metric Counter for Slot 4 read transaction count
0x0750	Metric Counter 29(1)	Metric Counter for Slot 4 read latency count
0x0760	Metric Counter 30(1)	Metric Counter for Slot 5 write byte count
0x0770	Metric Counter 31(1)	Metric Counter for Slot 5 write transaction count
0x0780	Metric Counter 32(1)	Metric Counter for Slot 5 write latency count
0x0790	Metric Counter 33(1)	Metric Counter for Slot 5 read byte count
0x07A0	Metric Counter 34(1)	Metric Counter for Slot 5 read transaction count
0x07B0	Metric Counter 35(1)	Metric Counter for Slot 5 read latency count
0x0900	Metric Counter 36(1)	Metric Counter for Slot 6 write byte count
0x0910	Metric Counter 37(1)	Metric Counter for Slot 6 write transaction count
0x0920	Metric Counter 38(1)	Metric Counter for Slot 6 write latency count
0x0930	Metric Counter 39(1)	Metric Counter for Slot 6 read byte count
0x0940	Metric Counter 40(1)	Metric Counter for Slot 6 read transaction count
0x0950	Metric Counter 41(1)	Metric Counter for Slot 6 read latency count
0x0960	Metric Counter 42(1)	Metric Counter for Slot 7 write byte count
0x0970	Metric Counter 43(1)	Metric Counter for Slot 7 write transaction count
0x0980	Metric Counter 44(1)	Metric Counter for Slot 7 write latency count
0x0990	Metric Counter 45(1)	Metric Counter for Slot 7 read byte count
0x09A0	Metric Counter 46(1)	Metric Counter for Slot 7 read transaction count

Table 2-4: **Core Registers in Profile and Trace Modes *(Cont'd)***

Address Offset	Register Name	Description
0x09B0	Metric Counter 47(1)	Metric Counter for Slot 7 read latency count
0x0154	Metric Counter 48(1)	Metric Counter for Slot 0 maximum write and minimum write latency
0x0158	Metric Counter 49(1)	Metric Counter for Slot 0 maximum read and minimum read latency
0x01b4	Metric Counter 50(1)	Metric Counter for Slot 1 maximum write and minimum write latency
0x01b8	Metric Counter 51(1)	Metric Counter for Slot 1 maximum read and minimum read latency
0x0554	Metric Counter 52(1)	Metric Counter for Slot 2 maximum write and minimum write latency
0x0558	Metric Counter 53(1)	Metric Counter for Slot 2 maximum read and minimum read latency
0x05b4	Metric Counter 54(1)	Metric Counter for Slot 3 maximum write and minimum write latency
0x05b8	Metric Counter 55(1)	Metric Counter for Slot 3 maximum read and minimum read latency
0x0754	Metric Counter 56(1)	Metric Counter for Slot 4 maximum write and minimum write latency
0x0758	Metric Counter 57(1)	Metric Counter for Slot 4 maximum read and minimum read latency
0x07b4	Metric Counter 58(1)	Metric Counter for Slot 5 maximum write and minimum write latency
0x07b8	Metric Counter 59(1)	Metric Counter for Slot 5 maximum read and minimum read latency
0x0954	Metric Counter 60(1)	Metric Counter for Slot 6 maximum write and minimum write latency
0x0958	Metric Counter 61(1)	Metric Counter for Slot 6 maximum read and minimum read latency
0x09b4	Metric Counter 62(1)	Metric Counter for Slot 7 maximum write and minimum write latency
0x09b8	Metric Counter 63(1)	Metric Counter for Slot 7 maximum read and minimum read latency
Sampled Metric Counter Registers: Available in Profile Mode Only		
0x0200	Sampled Metric Counter 0(1)	Sampled Metric Counter for Slot 0 write byte count
0x0210	Sampled Metric Counter 1(1)	Sampled Metric Counter for Slot 0 write transaction count
0x0220	Sampled Metric Counter 2(1)	Sampled Metric Counter for Slot 0 write latency count

Table 2-4: **Core Registers in Profile and Trace Modes *(Cont'd)***

Address Offset	Register Name	Description
0x0230	Sampled Metric Counter 3(1)	Sampled Metric Counter for Slot 0 read byte count
0x0240	Sampled Metric Counter 4(1)	Sampled Metric Counter for Slot 0 read transaction count
0x0250	Sampled Metric Counter 5(1)	Sampled Metric Counter for Slot 0 read latency count
0x0260	Sampled Metric Counter 6(1)	Sampled Metric Counter for Slot 1 write byte count
0x0270	Sampled Metric Counter 7(1)	Sampled Metric Counter for Slot 1 write transaction count
0x0280	Sampled Metric Counter 8(1)	Sampled Metric Counter for Slot 1 write latency count
0x0290	Sampled Metric Counter 9(1)	Sampled Metric Counter for Slot 1 read byte count
0x02A0	Sampled Metric Counter 10(1)	Sampled Metric Counter for Slot 1 read transaction count
0x02B0	Sampled Metric Counter 11(1)	Sampled Metric Counter for Slot 1 read latency count
0x0600	Sampled Metric Counter 12(1)	Sampled Metric Counter for Slot 2 write byte count
0x0610	Sampled Metric Counter 13(1)	Sampled Metric Counter for Slot 2 write transaction count
0x0620	Sampled Metric Counter 14(1)	Sampled Metric Counter for Slot 2 write latency count
0x0630	Sampled Metric Counter 15(1)	Sampled Metric Counter for Slot 2 read byte count
0x640	Sampled Metric Counter 16(1)	Sampled Metric Counter for Slot 2 read transaction count
0x0650	Sampled Metric Counter 17(1)	Sampled Metric Counter for Slot 2 read latency count
0x0660	Sampled Metric Counter 18(1)	Sampled Metric Counter for Slot 3 write byte count
0x0670	Sampled Metric Counter 19(1)	Sampled Metric Counter for Slot 3 write transaction count
0x0680	Sampled Metric Counter 20(1)	Sampled Metric Counter for Slot 3 write latency count
0x0690	Sampled Metric Counter 21(1)	Sampled Metric Counter for Slot 3 read byte count
0x06A0	Sampled Metric Counter 22(1)	Sampled Metric Counter for Slot 3 read transaction count
0x06B0	Sampled Metric Counter 23(1)	Sampled Metric Counter for Slot 3 read latency count

Table 2-4: **Core Registers in Profile and Trace Modes *(Cont'd)***

Address Offset	Register Name	Description
0x0800	Sampled Metric Counter 24(1)	Sampled Metric Counter for Slot 4 write byte count
0x0810	Sampled Metric Counter 25(1)	Sampled Metric Counter for Slot 4 write transaction count
0x0820	Sampled Metric Counter 26(1)	Sampled Metric Counter for Slot 4 write latency count
0x0830	Sampled Metric Counter 27(1)	Sampled Metric Counter for Slot 4 read byte count
0x0840	Sampled Metric Counter 28(1)	Sampled Metric Counter for Slot 4 read transaction count
0x0850	Sampled Metric Counter 29(1)	Sampled Metric Counter for Slot 4 read latency count
0x0860	Sampled Metric Counter 30(1)	Sampled Metric Counter for Slot 5 write byte count
0x0870	Sampled Metric Counter 31(1)	Sampled Metric Counter for Slot 5 write transaction count
0x0880	Sampled Metric Counter 32(1)	Sampled Metric Counter for Slot 5 write latency count
0x0890	Sampled Metric Counter 33(1)	Sampled Metric Counter for Slot 5 read byte count
0x08A0	Sampled Metric Counter 34(1)	Sampled Metric Counter for Slot 5 read transaction count
0x08B0	Sampled Metric Counter 35(1)	Sampled Metric Counter for Slot 5 read latency count
0x0A00	Sampled Metric Counter 36(1)	Sampled Metric Counter for Slot 6 write byte count
0x0A10	Sampled Metric Counter 37(1)	Sampled Metric Counter for Slot 6 write transaction count
0x0A20	Sampled Metric Counter 38(1)	Sampled Metric Counter for Slot 6 write latency count
0x0A30	Sampled Metric Counter 39(1)	Sampled Metric Counter for Slot 6 read byte count
0x0A40	Sampled Metric Counter 40(1)	Sampled Metric Counter for Slot 6 read transaction count
0x0A50	Sampled Metric Counter 41(1)	Sampled Metric Counter for Slot 6 read latency count
0x0A60	Sampled Metric Counter 42(1)	Sampled Metric Counter for Slot 7 write byte count
0x0A70	Sampled Metric Counter 43(1)	Sampled Metric Counter for Slot 7 write transaction count
0x0A80	Sampled Metric Counter 44(1)	Sampled Metric Counter for Slot 7 write latency count

Table 2-4: **Core Registers in Profile and Trace Modes *(Cont'd)***

Address Offset	Register Name	Description
0x0A90	Sampled Metric Counter 45(1)	Sampled Metric Counter for Slot 7 read byte count
0x0AA0	Sampled Metric Counter 46(1)	Sampled Metric Counter for Slot 7 read transaction count
0x0AB0	Sampled Metric Counter 47(1)	Sampled Metric Counter for Slot 7 read latency count
0x0254	Sampled Metric Counter 48(1)	Sampled Metric Counter for Slot 0 maximum write and minimum write latency
0x0258	Sampled Metric Counter 49(1)	Sampled Metric Counter for Slot 0 maximum read and minimum read latency
0x02b4	Sampled Metric Counter 50(1)	Sampled Metric Counter for Slot 1 maximum write and minimum write latency
0x02b8	Sampled Metric Counter 51(1)	Sampled Metric Counter for Slot 1 maximum read and minimum read latency
0x0654	Sampled Metric Counter 52(1)	Sampled Metric Counter for Slot 2 maximum write and minimum write latency
0x0658	Sampled Metric Counter 53(1)	Sampled Metric Counter for Slot 2 maximum read and minimum read latency
0x06b4	Sampled Metric Counter 54(1)	Sampled Metric Counter for Slot 3 maximum write and minimum write latency
0x06b8	Sampled Metric Counter 55(1)	Sampled Metric Counter for Slot 3 maximum read and minimum read latency
0x0854	Sampled Metric Counter 56(1)	Sampled Metric Counter for Slot 4 maximum write and minimum write latency
0x0858	Sampled Metric Counter 57(1)	Sampled Metric Counter for Slot 4 maximum read and minimum read latency
0x08b4	Sampled Metric Counter 58(1)	Sampled Metric Counter for Slot 5 maximum write and minimum write latency
0x08b8	Sampled Metric Counter 59(1)	Sampled Metric Counter for Slot 5 maximum read and minimum read latency
0x0a54	Sampled Metric Counter 60(1)	Sampled Metric Counter for Slot 6 maximum write and minimum write latency
0x0a58	Sampled Metric Counter 61(1)	Sampled Metric Counter for Slot 6 maximum read and minimum read latency
0x0ab4	Sampled Metric Counter 62(1)	Sampled Metric Counter for Slot 7 maximum write and minimum write latency
0x0ab8	Sampled Metric Counter 63(1)	Sampled Metric Counter for Slot 7 maximum read and minimum read latency
Event Log Registers: Available in Trace Mode Only		
0x0404	Software-written Data Register(2)	Software-written Data Register

Table 2-4: **Core Registers in Profile and Trace Modes *(Cont'd)***

Address Offset	Register Name	Description

Notes:
1. Available in profile mode only.
2. Available in trace mode only.

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 2-5.

Table 2-5: **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. This register is active only in advanced mode.
2. Because the S_AXI data width is 32 bits, the register can be read by a read transaction to two locations: (0x0000) and (0x0004).
3. If Global Count Width is 32, only the lower 32 bits of the register are valid.
4. 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-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 2-6.

Table 2-6: **Sample Interval Register Bit Definitions (0x0024)**

Bits	Name	Access	Reset Value	Description
31-0	Sample Interval	Read / Write	0x00	Sample Interval Register

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 2-7.

Table 2-7: **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	Read/Write	1	1: Resets metric counters when sample interval timer expires or when the sample register is read. 0: Metric Counters are not reset when sample interval counter lapses or when the sample register is read.
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.

Sample Register

The sample register read captures metric counts and stores the values in the sample metric counters. The value in this register is a free-running counter value in terms of AXI4-Lite clocks. The bit definition and accessibility of this register is shown in Table 2-8.

Table 2-8: **Sample Register Bit Definitions (0x002C)**

Bits	Name	Access	Reset Value	Description
31-0	Sample Register Read	Read	0	Free running counter values.

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 2-9: **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 2-10: **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. Valid only in advanced mode. 1: Enabled 0: Disabled
11	Metric Counter 8 Overflow Interrupt Enable	Read/Write	0	Metric Counter 8 Overflow Interrupt. Valid only in advanced mode. 1: Enabled 0: Disabled
10	Metric Counter 7 Overflow Interrupt Enable	Read/Write	0	Metric Counter 7 Overflow Interrupt. Valid only in advanced mode. 1: Enabled 0: Disabled
9	Metric Counter 6 Overflow Interrupt Enable	Read/Write	0	Metric Counter 6 Overflow Interrupt. Valid only in advanced mode. 1: Enabled 0: Disabled
8	Metric Counter 5 Overflow Interrupt Enable	Read/Write	0	Metric Counter 5 Overflow Interrupt . Valid only in advanced mode. 1: Enabled 0: Disabled
7	Metric Counter 4 Overflow Interrupt Enable	Read/Write	0	Metric Counter 4 Overflow Interrupt. Valid only in advanced mode. 1: Enabled 0: Disabled
6	Metric Counter 3 Overflow Interrupt Enable	Read/Write	0	Metric Counter 3 Overflow Interrupt. Valid only in advanced mode. 1: Enabled 0: Disabled

Table 2-10: **Interrupt Enable Register Bit Definitions (0x0034) *(Cont'd)***

Bits	Name	Access	Reset Value	Description
5	Metric Counter 2 Overflow Interrupt Enable	Read/Write	0	Metric Counter 2 Overflow Interrupt. Valid only in advanced mode. 1: Enabled 0: Disabled
4	Metric Counter 1 Overflow Interrupt Enable	Read/Write	0	Metric Counter 1 Overflow Interrupt. Valid only in advanced mode. 1: Enabled 0: Disabled
3	Metric Counter 0 Overflow Interrupt Enable	Read/Write	0	Metric Counter 0 Overflow Interrupt. Valid only in advanced mode. 1: Enabled 0: Disabled
2	Event Log FIFO Overflow Interrupt Enable	Read/Write	0	Event Log FIFO Overflow Interrupt. Valid in advanced mode (when event logging is enabled) and trace modes.
1	Sample Interval Counter Overflow Interrupt Enable	Read/Write	0	Sample Interval Counter Overflow Interrupt. Valid in advanced and profile mode. 1: Enabled 0: Disabled
0	Global Clock Counter Overflow Interrupt Enable	Read/Write	0	Global Clock Counter Overflow Interrupt. Valid only in advanced mode. 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 2-11: **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. Valid only in advanced mode. 1: Active 0: Not Active
11	Metric Counter 8 Overflow Interrupt	Read/Write	0	Metric Counter 8 Overflow Interrupt. Valid only in advanced mode. 1: Active 0: Not Active
10	Metric Counter 7 Overflow Interrupt	Read/Write	0	Metric Counter 7 Overflow Interrupt. Valid only in advanced mode. 1: Active 0: Not Active
9	Metric Counter 6 Overflow Interrupt	Read/Write	0	Metric Counter 6 Overflow Interrupt. Valid only in advanced mode. 1: Active 0: Not Active
8	Metric Counter 5 Overflow Interrupt	Read/Write	0	Metric Counter 5 Overflow Interrupt. Valid only in advanced mode. 1: Active 0: Not Active
7	Metric Counter 4 Overflow Interrupt	Read/Write	0	Metric Counter 4 Overflow Interrupt. Valid only in advanced mode. 1: Active 0: Not Active
6	Metric Counter 3 Overflow Interrupt	Read/Write	0	Metric Counter 3 Overflow Interrupt. Valid only in advanced mode. 1: Active 0: Not Active
5	Metric Counter 2 Overflow Interrupt	Read/Write	0	Metric Counter 2 Overflow Interrupt. Valid only in advanced mode. 1: Active 0: Not Active

Table 2-11: **Interrupt Status Register Bit Definitions (0x0038) *(Cont'd)***

Bits	Name	Access	Reset Value	Description
4	Metric Counter 1 Overflow Interrupt	Read/Write	0	Metric Counter 1 Overflow Interrupt. Valid only in advanced mode. 1: Active 0: Not Active
3	Metric Counter 0 Overflow Interrupt	Read/Write	0	Metric Counter 0 Overflow Interrupt. Valid only in advanced mode. 1: Active 0: Not Active
2	Event Log FIFO Overflow	Read/Write	0	Event Log FIFO Overflow Interrupt . Valid in advanced mode (when event logging is enabled) and trace mode. 1: Active 0: Not Active
1	Sample Interval Counter Overflow Interrupt	Read/Write	0	Sample Interval Counter Overflow Interrupt. Valid in advanced and profile mode. 1: Active 0: Not Active
0	Global Clock Counter Overflow Interrupt	Read/Write	0	Global Clock Counter Overflow Interrupt. Valid only in advanced mode. 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 2-12.

Table 2-12: **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.

Table 2-12: **Metric Selector Register 0 Bit Definitions (0x0044) *(Cont'd)***

Bits	Name	Access	Reset Value	Description
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. This register is valid only when event counters are enabled in advanced mode.
2. The number of Metric Counters that exist in the system is based on the No of Event Counters parameter.
3. 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 2-13.

Table 2-13: **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. This register is valid only when event counters are enabled in advanced mode.
2. The number of Metric Counters that exist in the system is based on the No of Event Counters parameter.
3. 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 2-14.

Table 2-14: **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. This register is valid only when event counters are enabled in advanced mode.
2. The number of Metric Counters that exist in the system is based on the No of Event Counters parameter.
3. 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 2-15 and Table 2-16.

Table 2-15: **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. Count increments for every write address acceptance on the interface.
1	Read Transaction Count	Number of read transactions by/to a particular master/ slave. Count increments for every read address acceptance on the interface.
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.

Table 2-15: **Metric Descriptions *(Cont'd)***

Metric Selector Value	Metric	Description
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. The default minimum write latency provided by the core is 0xFFFFFFFF.
13	Maximum Write Latency	Maximum write latency number.
14	Minimum Read Latency	Minimum Read Latency number. The default minimum read latency provided by the core is 0xFFFFFFFF.
15	Maximum Read Latency	Maximum Read latency number.
Metrics Computed for AXI4-Stream Agent		
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 2-16: **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.

Table 2-16: **Metric Slot Descriptions**

Metric Slot ID Value	Slot	Description
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 2-17.

Table 2-17: **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 the Vivado IDE.
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.
4. Metric Counters 0 to 47 are valid in advanced and profile modes.
5. Metric Counters 48 to 63 are only valid in profile mode.
6. Metric Counters 48 to 63 contain the minimum or maximum latency numbers. They are not accumulated values.
7. The MSB 16 bits of Metric Counters 48 to 63 contain the maximum latency number, and the LSB 16 bits of Metric Counters 48 to 63 contain the minimum latency number.

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 2-18. 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 2-18: **Incrementer Register Bit Definitions (0x0104)**

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

Notes:
1. Valid only when the event counters are enabled in advanced mode.
2. Number of Incrementers that exist in the system is based on No of Event Counter parameter set in Vivado IDE.
3. x varies from 0 to No of Event Counters-1.
4. 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 2-19.

Table 2-19: **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. Valid only when event counters are enabled in advanced mode.
2. Number of Range Registers that exist in the system is based on No of Event Counters parameter set in the Vivado IDE.
3. x varies from 0 to No Event Counters-1.
4. 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 2-20.

Table 2-20: **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.

Notes:
1. Valid only in advanced mode when both event counters and event logging is enabled.
2. Number of Metric Count Log Enable Registers that exist in the system is based on No of Event Counters parameter set in the Vivado IDE.
3. x varies from 0 to No of Event Counters-1.
4. 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 or sample register read. These registers are in AXI clock domain. The bit definition and accessibility of this register are shown in Table 2-21.

Table 2-21: **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 Vivado IDE.
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.
4. Sample Metric Counter 0 to 47 are valid in advanced and profile mode.
5. Sample Metric Counter 48 to 63 are only valid in profile mode.
6. Sample Metric Counters 48 to 63 contain the minimum or maximum latency numbers. They are not accumulated values.
7. The MSB 16 bits of Sample Metric Counters 48 to 63 contain the maximum latency number, and the LSB 16 bits of Sample Metric Counters 48 to 63 contain the minimum latency number.

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 or sample register read. 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 Incrementer Register Bit Definitions (0x0204)**

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

Notes:

1. Valid only when event counting is enabled in advanced mode.
2. Number of sampled incrementers that exist in the system is based on No of Event Counters parameter set in the Vivado IDE.
3. x varies from 0 to No of Event Counters-1.
4. Refer to the register space table for addresses of all the sampled incrementers.

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 2-23.

Table 2-23: **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 When the trigger parameter (C_EN_TRIGGER) is asserted, the same bit is also used to reset the complete trigger path logic.
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. Advanced mode only.
16	Global_Clk_Cnt_En	Read/Write	0	1: Enables the free-running Global Clock Counter. Advanced mode only.
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	Read Latency End Point	Read/Write	0	1: Enables first read as read latency end point 0 :Enables last read as read latency end point
6	Read Latency Start Point	Read/Write	0	0: Enables address issuance by the master interface as read latency start point (ARVALID) 1: Enables address acceptance by slave as read latency start point (ARVALID and ARREADY)

Table 2-23: **Control Register Bit Definitions (0x0300) *(Cont'd)***

Bits	Name	Access	Reset Value	Description
5	Write Latency End Point	Read/Write	0	1: Enables first write as write latency end point 0:Enables Last write as write latency end point
4	Write Latency Start Point	Read/Write	0	0: Enables address issuance by master interface as write latency start point (AWVALID) 1: Enables address acceptance by the slave interface as write latency start point (AWVALID and AWREADY)
3	Enable ID Based Filtering/Masking	Read/Write	0	This bit is only valid in Advanced mode. 0: Ignore ID for metric calculation 1: Enables ID filtering and masking. When enabled, all metric corresponds to the ID configured in the IDR and IDMR registers.
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 ***Note:*** De-assertion of metric reset (asserted though Control register) takes five clock cycles by core.
0	Metrics_Cnt_En	Read/Write	0	1: Enables all metric counters in the monitor

ID Register (IDR)

The write ID and read ID to be filtered/masked for the metric calculation to be configured in the register. These ID bits are used when ID-based filtering is enabled in the APM core. The bit definition and accessibility of this register are shown in Table 2-24.

Table 2-24: **ID Register (IDR) Bit Definitions (0x304)**

Bits	Name	Access	Reset Value	Description
31-16	RID	Read/Write	0x00	Read ID to Filter/Mask
15-0	WID	Read/Write	0x00	Write ID to Filter/Mask

Notes:

1. Valid only when event counting is enabled in advanced mode.

ID Mask Register (IDMR)

The mask to the write/read ID for all metric calculation must be configured in the register. The bit definition and accessibility of this register are shown in Table 2-25. When the mask bit is set, the APM ignores the corresponding bits on the interface for ID filtering.

Table 2-25: **ID Mask Register (IDMR) Bit Definitions (0x308)**

Bits	Name	Access	Reset Value	Description
31-16	RID	Read/Write	0x00	Read ID to Mask
15-0	WID	Read/Write	0x00	Write ID to Mask

Notes:

1. Valid only when event counting is enabled in advanced mode.

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

Notes:

1. Valid only when event counting is enabled in advanced mode. In trace mode, the flags are controlled through the parameters.

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

Notes:
1. Valid only when event counting is enabled in advanced mode or trace mode.

Designing with the Core

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

General Design Guidelines

Xilinx recommends the following steps for all designs using the AXI Performance Monitor core for performance counting and event logging.

Performance Counting using Event Count 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.

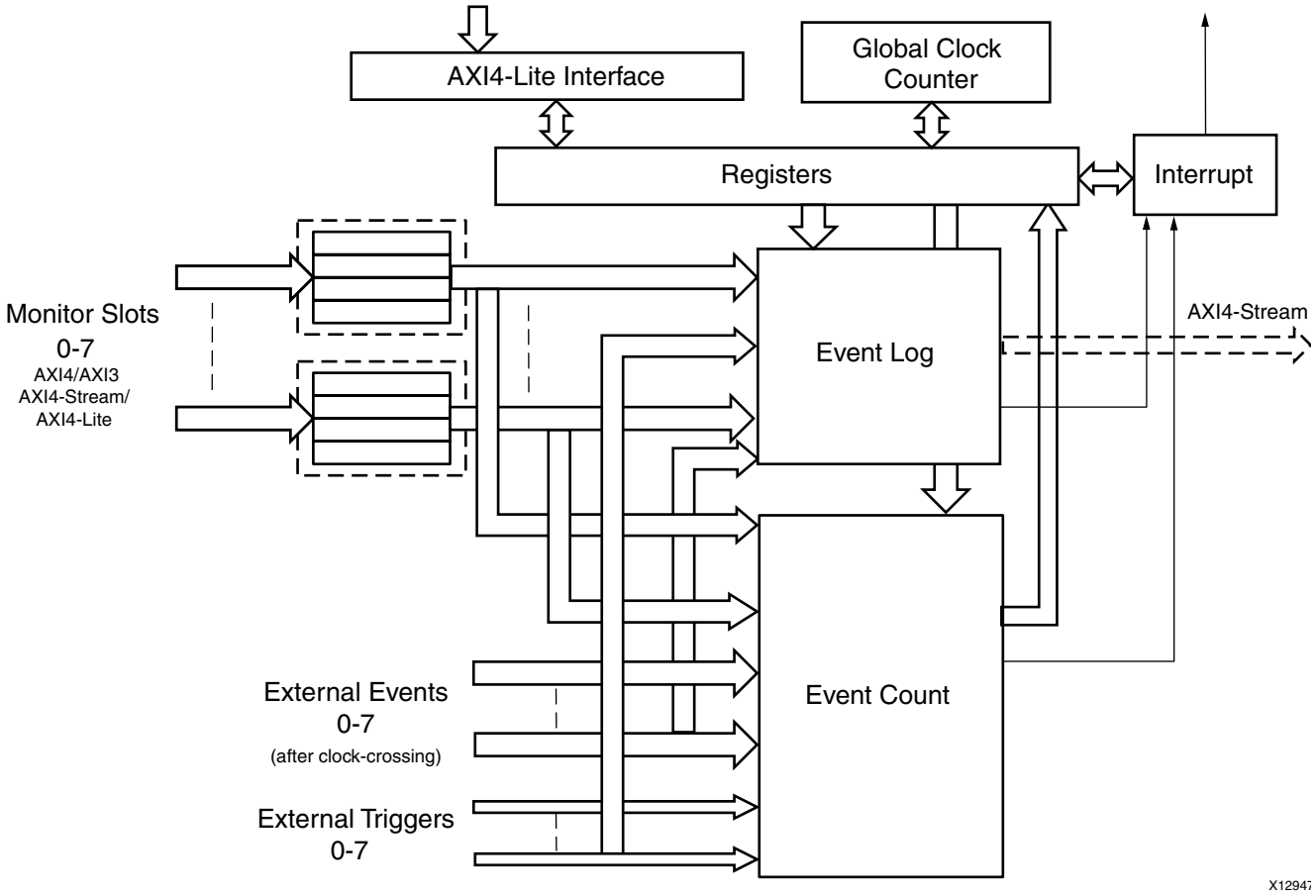

Figure 3-1: **Block Diagram of AXI Performance Monitor**

Event Logging using Event Log Module

1. Instantiate the AXI 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 required flags to capture events in the Event Log through the Flag Enable register.
6. Enable the Event Log bit in Control register.
7. Cross probing is optional and required to automatically start event log (configured using the enable event log bit in the Control register). 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.

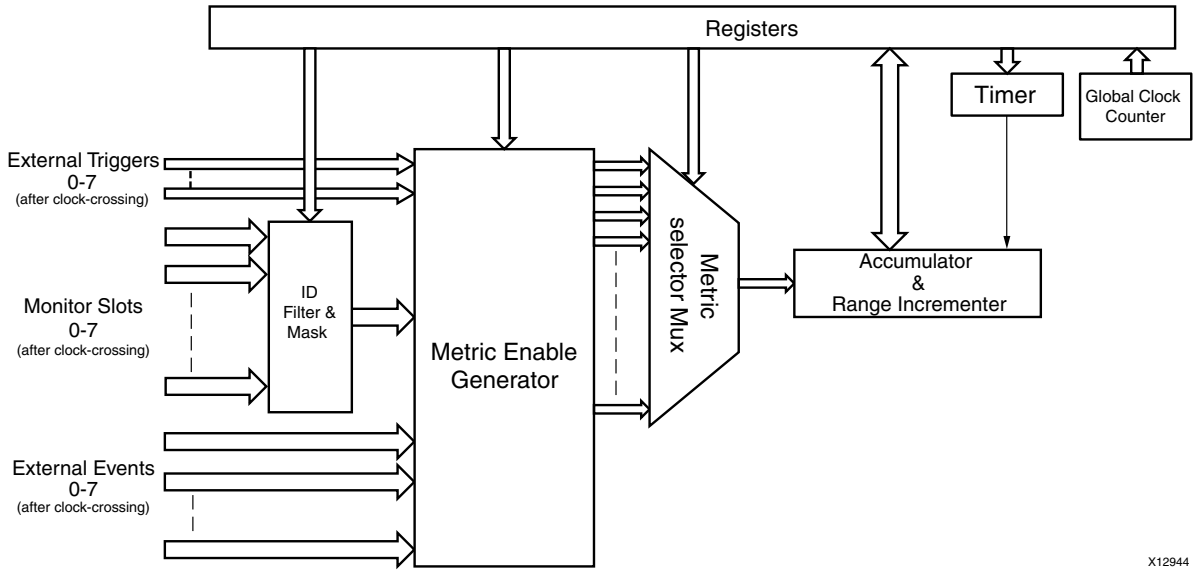

Figure 3-2: **Event Count Module**

Programming Sequence

This section describes the programming sequences for the available modes.

Advanced Mode

This section describes how to program the AXI Performance Monitor core for event counting and event logging in advanced mode.

Capturing Non-ID Based 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 0x00010001 in control register(0x300).

The Performance Monitor configuration is complete now, and the relevant slot transactions are 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,.....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,.....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 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,..... 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 after 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.

Capturing Metric Counts with Sample Register Read

This section uses a system built with two monitor slots, two external events, and three counters. In this example system, slot 1 is connected to the AXI4-Stream interface, and slot 2 is connected to the memory map interface. In this example, the steps to capture the metric counts with the sample register are as follows:

1. Reset the metric counters by writing 0x0000002 into the Control register (0x300).
2. Configure the metric selection registers (0x44) 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 0x16111210
        - Metric Counter 0: Slot 1 Transfer Cycle count (10)
        - Metric Counter 1: Slot 1 Data Byte count (12)

- Metric Counter 2: Slot 1 Packet count (11)
- Metric Counter 3: Slot 1 Master Idle count (16)

3. Enable metric counters by writing 0x00000001 in Control register (0x300). The AXI Performance Monitor core configuration is complete now, and the relevant slot transactions are active (slot 1 for this example).
4. Wait for the time to complete the transactions and read the Sample register (0x002C)
5. Read the sampled metric counters (0x200, 0x210 and 0x220) to capture the configured metrics.

Metric Counts for DMA/Memory Transfers with ID filtering

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

1. Reset the metric counters by writing 0x0000002 into the Control Register (0x300).
2. Configure the metric selection registers (0x44, 0x48, 0x4C) for the required slot metrics.
3. Configure the ID register (0x0304) with the write and read transaction ID to be filtered for metric calculation.
4. Configure the ID Mask register (0x0308) if there is any ID mask to be applied for ID filter. If the mask bit is '1' the corresponding bits of the transaction ID can be ignored in the ID filter.
5. Enable the metric counters by writing 0x00000001 in the Control Register (0x300).
6. Start DMA/Memory transfers.
7. After the transfers are complete, read the metric counters (0x100, 0x110,....up to 0x190) to get the configured metrics.

Profile Mode

This section describes the programming steps used in profile mode.

Capturing Metric Counts with Sample Register Read

This section uses a system built with two monitor slots and APM enabled with profile mode. Because profile mode supports only AXI4/AXI3/AXI4-Lite interfaces, slot 1 and slot 2 are connected to the AXI4 interface. In this example, the steps to capture the metric counts with the sample register are as follows:

1. Reset the metric counters by writing 0x0000002 into the Control register (0x300).
2. Enable Metric Counters by writing 0x00000001 in Control register (0x300). The AXI Performance Monitor core configuration is complete now, and the relevant slot 1 and slot 2 transactions are active.

3. Wait for the time to complete the transactions and read the Sample register (0x002C).
4. Read the sampled metric counters 0x200, 0x210, 0x220, 0x230, 0x240, 0x250 for slot 1 fixed metrics and 0x260, 0x270, 0x280, 0x290, 0x2A0 and 0x2B0 for slot 2 fixed metrics.

Trace Mode

This use case assumes that the AXI Performance Monitor core is enabled with trace mode and the following corresponding parameters are set:

- FIFO Depth
- Write Address Flag
- First Write Flag
- Last Write Flag
- Write Response Flag
- Read Address Flag
- First Read Flag
- Last Read Flag
- Software Register Write Flag
- Enable IDs In Log data
- Enable Transaction Length in Log data

The following steps capture the metric counts in trace mode:

1. Reset the AXI4-Stream FIFO by writing 0x2000000 into the Control register (0x300).
2. Enable the event log bit in the Control register (write 0x100 in 0x300).
3. Write to the Software-Written Data register with the data required to log.
4. Initiate AXI traffic over connected slots of APM.
5. As the flags are enabled with static configuration of the core, the APM starts logging when the requested flag occurs or when the data register data is written.
6. Offload the log data through the AXI4-Stream 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 domain, as follows:

- AXI4-Lite clock domain is reset by `s_axi_aresetn`.
- Core clock domain is reset by `core_aresetn`.
- AXI4/AXI3/AXI4-Lite slot clock domain is reset by `slot_n_axi_aresetn`.
- AXI4-Stream slot clock domain is reset by `slot_n_axis_aresetn`.
- External Event clock domain is reset by `ext_rstn_n`.
- Event log clock domain is reset by `m_axis_aresetn`.

Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 9]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 4]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 6]

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.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu .

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 4] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5].

Note: Figures in this chapter are illustrations of the Vivado® IDE. This layout might vary from the current version.

If you are customizing and generating the core in the IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 9] for detailed information. Vivado IDE might auto-compute certain configuration values when validating or generating the design, as noted in this section.

TIP: *You can view the parameter values after successful completion of the* `validate_bd_design` *command.*

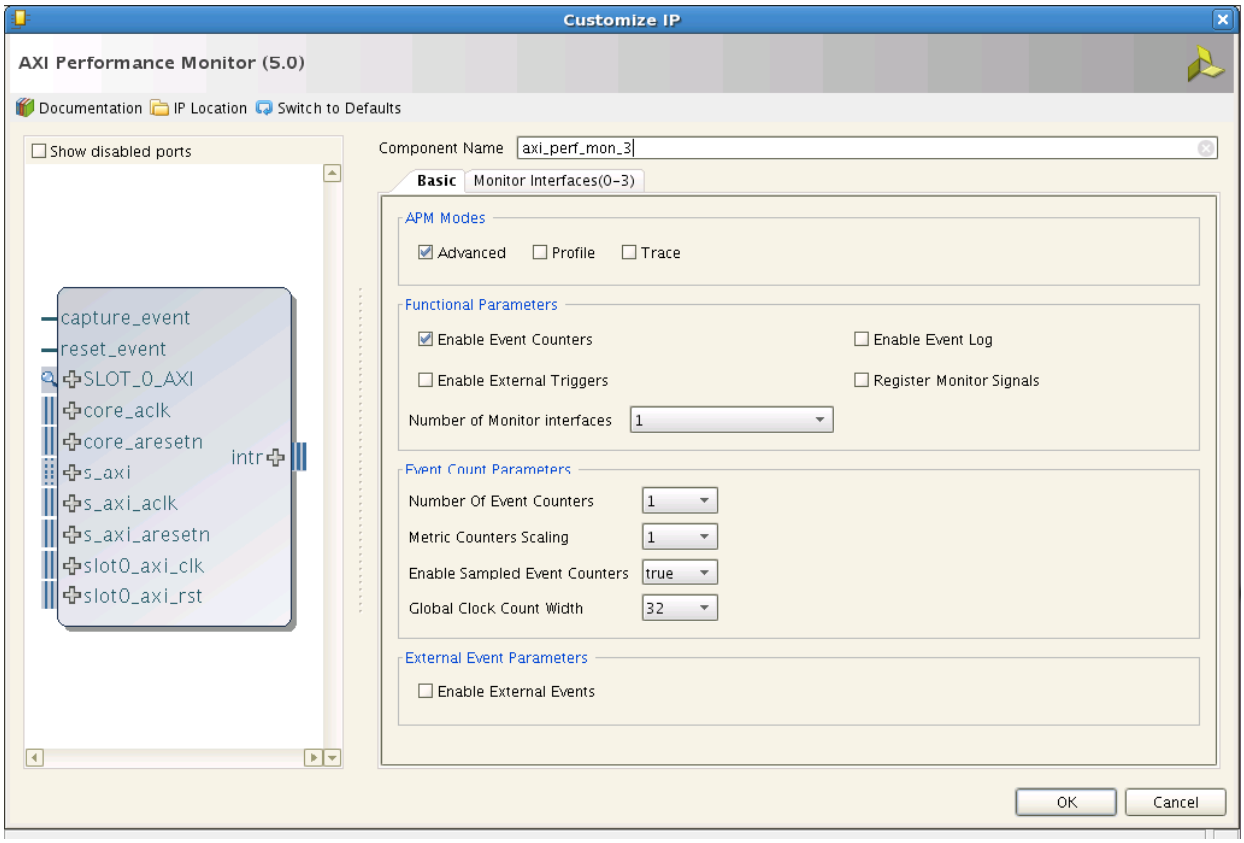

Figure 4-1: **Vivado IDE/IP Integrator Basic Tab**

The IDE 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 IDE options for the AXI Performance Monitor core.

APM Modes

- **Advanced**: Enables advanced mode.

- **Profile**: Enables profile mode.
- **Trace**: Enables trace mode.

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. In profile and trace modes, you cannot select the triggers manually. These are enabled by default.
- **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.
- **Metric Counters Scaling**: The actual metric counter values are scaled down based on the selected scaling factor. Allowed values are 1, 2, 4 and 8. The default scaling factor is 1.
- **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. In trace mode, you cannot select the external events manually. These are enabled by default.
- **Enable Synchronizers for Event (0 ... 7)**: Enables FIFO for synchronizing external event 0 signals to the core clock domain

  Note: When using Vivado IP integrator, the synchronizer parameters are automatically computed and set based on the connected clocks.

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.

Note: When using IP integrator, these the synchronizers are automatically computed and set based on the connected clocks.

- **ID Width**: Indicates the ID width of the AXI4/AXI3/AXI4-Lite slot.

  Note: When using IP integrator, the ID width for each slot is automatically computed and set based on the connected interface ID width. When using the AXI4-Lite interface, the ID width is set to 0.

- **Data Width**: Indicates the DATA width of the AXI4/AXI3/AXI4-Lite slot.

  Note: When using IP integrator, the data width for each slot is automatically computed and set based on the connected interface data width.

- **Address Width**: Indicates the Address width of the AXI4/AXI3/AXI4-Lite slot.

  Note: When using IP integrator, the address width for each slot is automatically computed and set based on the connected interface address width.

- **TUSER Width**: Indicates the ID width of the AXI4-Stream slot.

  Note: When using IP integrator, the TUSER width for each slot is automatically computed and set based on the connected interface TUSER width.

- **TDATA Width**: Indicates the data width of the AXI4-Stream slot.

  Note: When using IP integrator, the TDATA width for each slot is automatically computed and set based on the connected interface TDATA width.

- **TDEST Width**: Indicates the TDEST width of the AXI4-Stream slot.

  Note: When using IP integrator, the TDEST width for each slot is automatically computed and set based on the connected interface TDEST width.

- **TID Width**: Indicates the TID width of the AXI4-Stream slot.

  Note: When using IP integrator, the TID width for each slot is automatically computed and set based on the connected interface TID width.

Log Parameters

Event Log Streaming Parameters

- **Use Synchronous FIFO**: Enables the Synchronous FIFO. By default the core uses Asynchronous FIFO.

  Note: When using IP integrator, the synchronous FIFO for each slot is automatically computed and set based on the connected clocks.

- **Depth**: Event Log Streaming FIFO depth
- **TID Width**: Event Log Streaming TID Width

AXI4 Monitor Slot Parameters

- **Write Address Flag**: Captures the write address flags of the connected AXI slots in trace mode.

- **First Write Flag**: Captures the first write flags of the connected AXI slots in trace mode.
- **Last Write Flag**: Captures the last write flags of the connected AXI slots in trace mode.
- **Write Response Flag**: Captures the write response flags of the connected AXI slots in trace mode.
- **Read Address Flag**: Captures the read address flags of the connected AXI slots in trace mode.
- **First Read Flag**: Captures the first read flags of the connected AXI slots in trace mode.
- **Last Read Flag**: Captures the last read flags of the connected AXI slots in trace mode.
- **SW Register Write Flag**: Captures the SW register write flags of the connected AXI slots in trace mode.
- **External Event Flag**: Captures the external events flags of the connected AXI slots in trace mode.
- **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.

User Parameters

Table 4-1 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl console).

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter/Value[(1)]	User Parameter/Value[(1)]	Default Value
Streaming Interface TID width	C_FIFO_AXIS_TID_WIDTH	1
Streaming FIFO Depth	C_FIFO_AXIS_DEPTH	32
ID width	S_AXI_OFFLD_ID_WIDTH	1
Enable Event Log	C_ENABLE_EVENT_LOG	0
Enable Synchronizers for Event0	C_EXT_EVENT0_FIFO_ENABLE	1
Enable Synchronizers for Event1	C_EXT_EVENT1_FIFO_ENABLE	1
Enable Synchronizers for Event2	C_EXT_EVENT2_FIFO_ENABLE	1
Enable Synchronizers for Event3	C_EXT_EVENT3_FIFO_ENABLE	1
Enable Synchronizers for Event4	C_EXT_EVENT4_FIFO_ENABLE	1

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

Vivado IDE Parameter/Value[1]	User Parameter/Value[1]	Default Value
Enable Synchronizers for Event5	C_EXT_EVENT5_FIFO_ENABLE	1
Enable Synchronizers for Event6	C_EXT_EVENT6_FIFO_ENABLE	1
Enable Synchronizers for Event7	C_EXT_EVENT7_FIFO_ENABLE	1
Register All Monitor Signals	C_REG_ALL_MONITOR_SIGNALS	0
Slot 7:		
Enable Synchronizer	C_SLOT_7_FIFO_ENABLE	1
TUSER Width	C_SLOT_7_AXIS_TUSER_WIDTH	1
TDEST Width	C_SLOT_7_AXIS_TDEST_WIDTH	1
TID Width	C_SLOT_7_AXIS_TID_WIDTH	1
TDATA Width	C_SLOT_7_AXIS_TDATA_WIDTH	32
AXI Protocol	C_SLOT_7_AXI_PROTOCOL	AXI4
ID Width	C_SLOT_7_AXI_ID_WIDTH	1
Data Width	C_SLOT_7_AXI_DATA_WIDTH	32
Address Width	C_SLOT_7_AXI_ADDR_WIDTH	32
Slot 6:		
Enable Synchronizer	C_SLOT_6_FIFO_ENABLE	1
TUSER Width	C_SLOT_6_AXIS_TUSER_WIDTH	1
TDEST Width	C_SLOT_6_AXIS_TDEST_WIDTH	1
TID Width	C_SLOT_6_AXIS_TID_WIDTH	1
TDATA Width	C_SLOT_6_AXIS_TDATA_WIDTH	32
AXI Protocol	C_SLOT_6_AXI_PROTOCOL	AXI4
ID Width	C_SLOT_6_AXI_ID_WIDTH	1
Data Width	C_SLOT_6_AXI_DATA_WIDTH	32
Address Width	C_SLOT_6_AXI_ADDR_WIDTH	32
Slot 5:		
Enable Synchronizer	C_SLOT_5_FIFO_ENABLE	1
TUSER Width	C_SLOT_5_AXIS_TUSER_WIDTH	1
TDEST Width	C_SLOT_5_AXIS_TDEST_WIDTH	1
TID Width	C_SLOT_5_AXIS_TID_WIDTH	1
TDATA Width	C_SLOT_5_AXIS_TDATA_WIDTH	32
AXI Protocol	C_SLOT_5_AXI_PROTOCOL	AXI4
ID Width	C_SLOT_5_AXI_ID_WIDTH	1
Data Width	C_SLOT_5_AXI_DATA_WIDTH	32
Address Width	C_SLOT_5_AXI_ADDR_WIDTH	32
Slot 4:		
Enable Synchronizer	C_SLOT_4_FIFO_ENABLE	1
TUSER Width	C_SLOT_4_AXIS_TUSER_WIDTH	1
TDEST Width	C_SLOT_4_AXIS_TDEST_WIDTH	1

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

Vivado IDE Parameter/Value[1]	User Parameter/Value[1]	Default Value
TID Width	C_SLOT_4_AXIS_TID_WIDTH	1
TDATA Width	C_SLOT_4_AXIS_TDATA_WIDTH	32
AXI Protocol	C_SLOT_4_AXI_PROTOCOL	AXI4
ID Width	C_SLOT_4_AXI_ID_WIDTH	1
Data Width	C_SLOT_4_AXI_DATA_WIDTH	32
Address Width	C_SLOT_4_AXI_ADDR_WIDTH	32
Slot 3:		
Enable Synchronizer	C_SLOT_3_FIFO_ENABLE	1
TUSER Width	C_SLOT_3_AXIS_TUSER_WIDTH	1
TDEST Width	C_SLOT_3_AXIS_TDEST_WIDTH	1
TID Width	C_SLOT_3_AXIS_TID_WIDTH	1
TDATA Width	C_SLOT_3_AXIS_TDATA_WIDTH	32
AXI Protocol	C_SLOT_3_AXI_PROTOCOL	AXI4
ID Width	C_SLOT_3_AXI_ID_WIDTH	1
Data Width	C_SLOT_3_AXI_DATA_WIDTH	32
Address Width	C_SLOT_3_AXI_ADDR_WIDTH	32
Slot 2:		
Enable Synchronizer	C_SLOT_2_FIFO_ENABLE	1
TUSER Width	C_SLOT_2_AXIS_TUSER_WIDTH	1
TDEST Width	C_SLOT_2_AXIS_TDEST_WIDTH	1
TID Width	C_SLOT_2_AXIS_TID_WIDTH	1
TDATA Width	C_SLOT_2_AXIS_TDATA_WIDTH	32
AXI Protocol	C_SLOT_2_AXI_PROTOCOL	AXI4
ID Width	C_SLOT_2_AXI_ID_WIDTH	1
Data Width	C_SLOT_2_AXI_DATA_WIDTH	32
Address Width	C_SLOT_2_AXI_ADDR_WIDTH	32
Slot 1:		
Enable Synchronizer	C_SLOT_1_FIFO_ENABLE	1
TUSER Width	C_SLOT_1_AXIS_TUSER_WIDTH	1
TDEST Width	C_SLOT_1_AXIS_TDEST_WIDTH	1
TID Width	C_SLOT_1_AXIS_TID_WIDTH	1
TDATA Width	C_SLOT_1_AXIS_TDATA_WIDTH	32
AXI Protocol	C_SLOT_1_AXI_PROTOCOL	AXI4
ID Width	C_SLOT_1_AXI_ID_WIDTH	1
Data Width	C_SLOT_1_AXI_DATA_WIDTH	32
Address Width	C_SLOT_1_AXI_ADDR_WIDTH	32
Slot 0:		
Enable Synchronizer	C_SLOT_0_FIFO_ENABLE	1

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

Vivado IDE Parameter/Value[1]	User Parameter/Value[1]	Default Value
TUSER Width	C_SLOT_0_AXIS_TUSER_WIDTH	1
TDEST Width	C_SLOT_0_AXIS_TDEST_WIDTH	1
TID Width	C_SLOT_0_AXIS_TID_WIDTH	1
TDATA Width	C_SLOT_0_AXIS_TDATA_WIDTH	32
AXI Protocol	C_SLOT_0_AXI_PROTOCOL	AXI4
ID Width	C_SLOT_0_AXI_ID_WIDTH	1
Data Width	C_SLOT_0_AXI_DATA_WIDTH	32
Address Width	C_SLOT_0_AXI_ADDR_WIDTH	32
Metric Counters Scaling	C_METRIC_COUNT_SCALE	1
Global Clock Count Width	C_GLOBAL_COUNT_WIDTH	32
Number of Counters	C_NUM_OF_COUNTERS	1
Enable Event Counters	C_ENABLE_EVENT_COUNT	1
Number of Monitor Slots	C_NUM_MONITOR_SLOTS	1
Is Event Log FIFO Synchronous	C_FIFO_AXIS_SYNC	0
Show AXI IDs In Log Data	C_SHOW_AXI_IDS	0
Show AXI Transaction Length in Log Data	C_SHOW_AXI_LEN	0
Show AXI Streaming TID In Log Data	C_SHOW_AXIS_TID	0
Show AXI Streaming TDEST	C_SHOW_AXIS_TDEST	0
Show AXI Streaming TUSER	C_SHOW_AXIS_TUSER	0
Enable External Events	ENABLE_EXT_EVENTS	0
Enable External Triggers	ENABLE_EXT_TRIGGERS	0
Advanced	C_ENABLE_ADVANCED	1
Profile	C_ENABLE_PROFILE	0
Trace	C_ENABLE_TRACE	0
Write Address Flag	C_EN_WR_ADD_FLAG	1
First Write Flag	C_EN_FIRST_WRITE_FLAG	1
Last Write Flag	C_EN_LAST_WRITE_FLAG	1
Write Response Flag	C_EN_RESPONSE_FLAG	1
Read Address Flag	C_EN_RD_ADD_FLAG	1
First Read Flag	C_EN_FIRST_READ_FLAG	1
Last Read Flag	C_EN_LAST_READ_FLAG	1
External Events Flag	C_EN_EXT_EVENTS_FLAG	0
Sw Register Write Flag	C_EN_SW_REG_WR_FLAG	0
Enable Trigger Logic	C_EN_TRIGGER	0
Enable Address Flags	C_EN_AXI_DEBUG	0

Notes:

1. Parameter values are listed in the table where the GUI parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.

Constraining the Core

This chapter contains information about constraining the core when using the Vivado® Design Suite.

The following constraints files are delivered in the `<project_name>/<project_name>.srcs/source_1/ip/<component_name>/` directory:

- `<component_name>.xdc`
- `<component_name>_clocks.xdc`

Core also delivers the following constraints for out-of-context (OOC) mode:

`<component_name>_ooc.xdc`

Required Constraints

There are no required constraints for this core.

Device, Package, and Speed Grade Selections

There are no device, package, or speed grade constraints for this core.

Clock Frequencies

In synchronous mode, all clocks run at the same frequency and are derived from the same source. There are no multicycle or false paths in this design. In asynchronous mode, the corresponding clocks are treated asynchronously to each other and the core writes out appropriate clock domain crossing constraints.

Clock Management

There are no clock management constraints for this core.

Clock Placement

There are no clock placement constraints for this core.

Banking

There are no banking constraints for this core.

Transceiver Placement

There are no transceiver placement constraints for this core.

I/O Standard and Placement

There are no I/O constraints for this core.

Simulation

This section contains information about simulating IP in the Vivado® Design Suite. For details, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4].

IMPORTANT: *For cores targeting 7 series or Zynq-7000 devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

Synthesis and Implementation

This section contains information about synthesis and implementation in the Vivado® Design Suite.

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 4].

Example Design

This chapter contains information about the example design provided in the Vivado® Design Suite.

Overview

The top module instantiates all components of the core and example design that are needed to implement the design in hardware, as shown in Figure 5-1. This includes clock generator, register configuration, data generator and data checker modules.

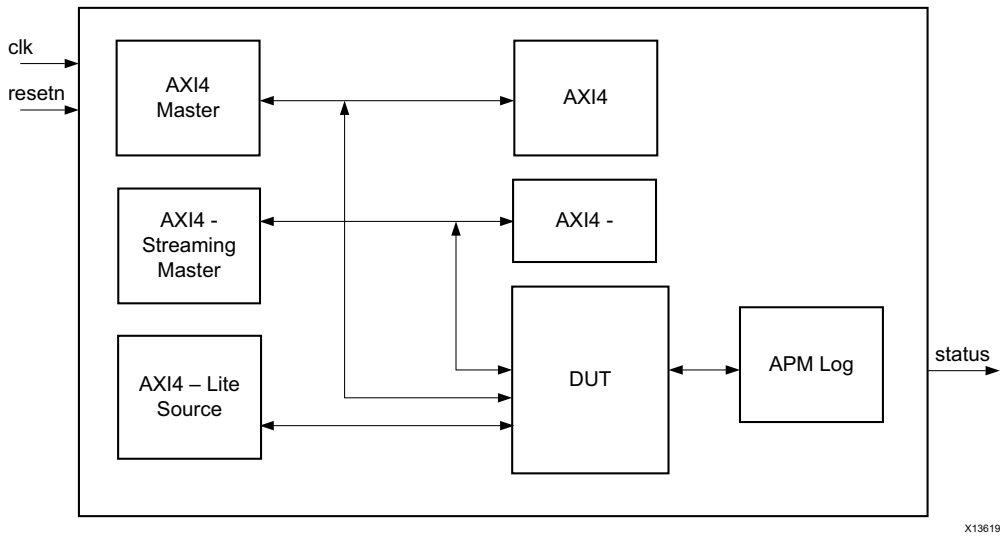

Figure 5-1: **Example Design Block Diagram**

This example design is to demonstrate transactions on AXI4-Lite, AXI4 and AXI4-Stream interfaces of the DUT and measure the metrics calculated by APM.

- **Clock Generator**: The clock is generated from the clock generator module.
- **AXI4-Lite Source**: This module configures the DUT registers as mentioned in the programming sequence. It reads and compares the metric counters to the expected values from the AXI4 generated stimulus.
- **AXI4 Master**: This module generates the AXI4 full traffic with a set length and burst size.

- **AXI4 BRAM**: This module is the slave to the AXI4 Master.
- **AXI4 Streaming Master**: This module generates the AXI4-Stream transactions with known number of packets
- **AXI4 Streaming Slave**: This logic generates the TREADY signal required by the AXI4-Stream master transaction.
- **APM Log Data Checker**: This module checks data received on the AXI4-Stream interface in case of event logging and trace modes.

Implementing the Example Design

After following the steps described in Chapter 3, Designing with the Core, implement the example design using the following instructions:

1. Right-click the core in the Hierarchy window, and select **Open IP Example Design**.
2. A new window will pop up, asking you to specify a directory for the example design. Select a new directory, or keep the default directory.
3. A new project is automatically created in the selected directory and will be opened in a new Vivado IDE window.
4. In the Flow Navigator (left side pane), click **Run Implementation** and follow the directions.

Example Design Directory Structure

In the current project directory, a new project named `<component_name>_example` is created and the files are generated in `<component_name>_example.src/sources_1/ip/<component_name>/` directory. This directory and its subdirectories contain all the source files that are required to create the AXI Performance Monitor example design.

The example_design directory is created in `<component_name>_example.src/sources_1/imports/<component_name>`. It contains the following generated example design top files:

- `<component_name>_exdes.xdc`: Top-level constraints file for the example design. The XDC delivered with the example design is configured for the KC705 board. The I/O constraints are commented by default. Un-comment them before implementing the example design on the KC705 board.
- `<component_name>_exdes.v`: Top-level HDL file for the example design.
- `apm_log_data.v`: Event Log checker module.
- `clock_gen.v`: Clock generation module.

Test Bench

This chapter contains information about the test bench provided in the Vivado® Design Suite.

The `<component_name>/simulation` directory contains the generated example design test bench files. The AXI Performance Monitor test bench generates a top-level 200 MHz clock and test pass/fail display message based on example design status output. Figure 6-1 shows a block diagram of the example design test bench.

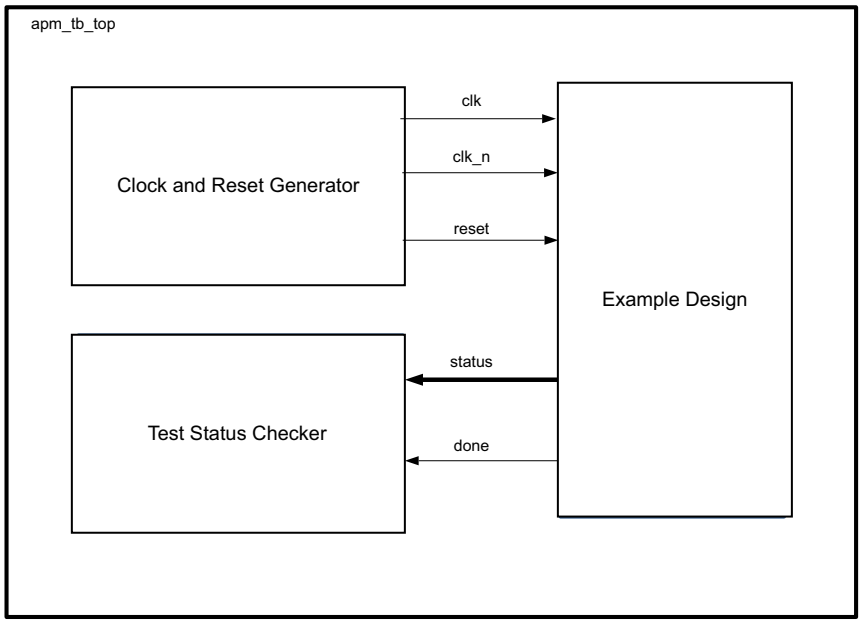

Figure 6-1: **Example Design Test Bench Block Diagram**

Simulating the Example Design

Using the AXI Performance Monitor example design (delivered as part of the AXI Performance Monitor), you can quickly simulate and observe the behavior of the AXI Performance Monitor. For details about simulating the core with Vivado Design Suite, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 6].

This section contains instructions for running a functional simulation of the AXI Performance Monitor example design. The example design supports functional (behavioral) and post-synthesis simulations.

- To run a functional simulation, click **Run Simulation** in the Flow Navigator (left pane) and then click **Run Behavioral Simulation**.
- To run a post-synthesis simulation, click **Run Simulation** in the Flow Navigator (left pane), and then click **Run Post-Synthesis Functional Simulation**.

Simulation Results

The simulation script compiles the AXI Performance Monitor example design, and supporting simulation files. It then runs the simulation and checks to ensure that it completed successfully.

If test fails, the following message is displayed:

```
Test Failed
```

If test passes, the following message is displayed:

```
Test Completed Successfully
```

Migrating and Upgrading

This appendix contains information about migrating a design from ISE® to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

For information about migrating to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 10].

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

Parameter Changes

Two new parameters were added for this release. These parameters are noted in Table 4-1.

Port Changes

Two new ports were added for this release to support trigger feature. These new ports are noted in Table 2-2.

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.

Finding Help on Xilinx.com

To help in the design and debug process when using the AXI Performance Monitor, the Xilinx Support web page (Xilinx Support web page) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

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 or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Technical Support

Xilinx provides technical support in the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change 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, navigate to the Xilinx Support web page.

Master Answer Record for AXI Performance Monitor

AR 54442

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 Design Suite Debug Feature

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

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.

Appendix C

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

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. *Vivado Design Suite User Guide: Getting Started* (UG910)
6. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
7. *Vivado Design Suite User Guide: Implementation* (UG904)
8. *Vivado Design Suite AXI Reference Guide* (UG1037)
9. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
10. *ISE to Vivado Design Suite Migration Methodology Guide* (UG911)
11. *Xilinx Software Development Kit User Guide: System Performance Analysis* (UG1145)
12. *System Performance Analysis of an All Programmable SoC* (XAPP1219)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/05/2016	5.0	• Added a note in Register Space section under Chapter 2, Product Specification. • Updated Automotive Applications Disclaimer in Please Read: Important Legal Notices section.
11/18/2015	5.0	• Added support for UltraScale+ architecture-based devices.
04/01/2015	5.0	• Enhanced the support for 64 bit address width. • Added the AXI debug and trigger features.
11/19/2014	5.0	• Clarified details for ID filtering in profile mode. • Added details about latency measurement in "Unsupported Features."
06/04/2014	5.0	Added Metric Counters 48 to 63 and Sample Metric Counters 48 to 63.
04/02/2014	5.0	• Added support for AXI4-Lite interface monitoring. • Updated the Maximum Frequency and Resource Utilization sections. • Marked core registers that only apply to profile and trace modes. • Consolidated Customizing and Generating the Core, Constraints, Simulation, and Synthesis information into new Design Flow Steps chapter.
12/18/2013	5.0	Added UltraScale™ architecture support.
10/02/2013	5.0	• Created three modes of operation: ◦ Advanced Mode ◦ Profile Mode ◦ Trace Mode • Added Example Design and Test Bench chapters. • Added IP integrator support. • Updated the Migrating and Upgrading appendix.
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 2-26.
03/20/2013	5.0	• Updated core to v4.0. • Removed support for Xilinx Platform Studio (XPS). • Added the Latency ID Register (LIDR).
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.
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.
07/25/2012	2.0	Added support for Vivado Design Suite. Added Event Log module and changed Event Count logic to be symmetric.
05/22/2012	1.1	Added Zynq-7000 support.
04/24/2012	1.0	Initial Xilinx release.

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.