PG307 AXI Sideband Formatter Utility v1.0

LogiCORE IP Product Guide

Vivado Design Suite

PG307 (v1.0) April 4, 2018

XILINX
ALL PROGRAMMABLE™

Table of Contents

IP Facts

The AXI Sideband Formatter Utility IP core inserts information into or recovers information from AXI USER signals for transport across a SmartConnect network.

Features

- **SMID for MPSoC SMMU:**
  - SMID insertion (optional)
  - SMID extraction (optional)
  - SMID removal (optional)
- **Parity:**
  - Generate parity on Write Data and Read Data channel output (optional)
  - Detect parity errors on Write Data and Read Data channel input (optional)

IP Facts Table

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	UltraScale+, UltraScale, 7 series and Zynq®-7000 AP SoC
Supported User Interfaces	AXI4 and AXI3
Resources	Not Applicable
Provided with Core	
Design Files	SystemVerilog
Example Design	N/A
Test Bench	N/A
Constraints File	N/A
Simulation Model	Unencrypted SystemVerilog

LogiCORE IP Facts Table	
Supported S/W Driver	N/A
Tested Design Flows[2]	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado
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. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The AXI Sideband Formatter Utility IP core inserts or recovers information into/from AXI User signals for transport across a SmartConnect network.

The core performs the following two functions.

- Transmission of SMID information along the AXI `awuser` and `aruser` signals.
- Generation, detection and transmission of parity information along the AXI `wuser` and `ruser` signals. Each instance of the IP can perform either or both of these two functions.

Feature Summary

SMID for MPSoC SMMU

- SMID insertion (optional)
- SMID extraction (optional)
- SMID removal (optional)

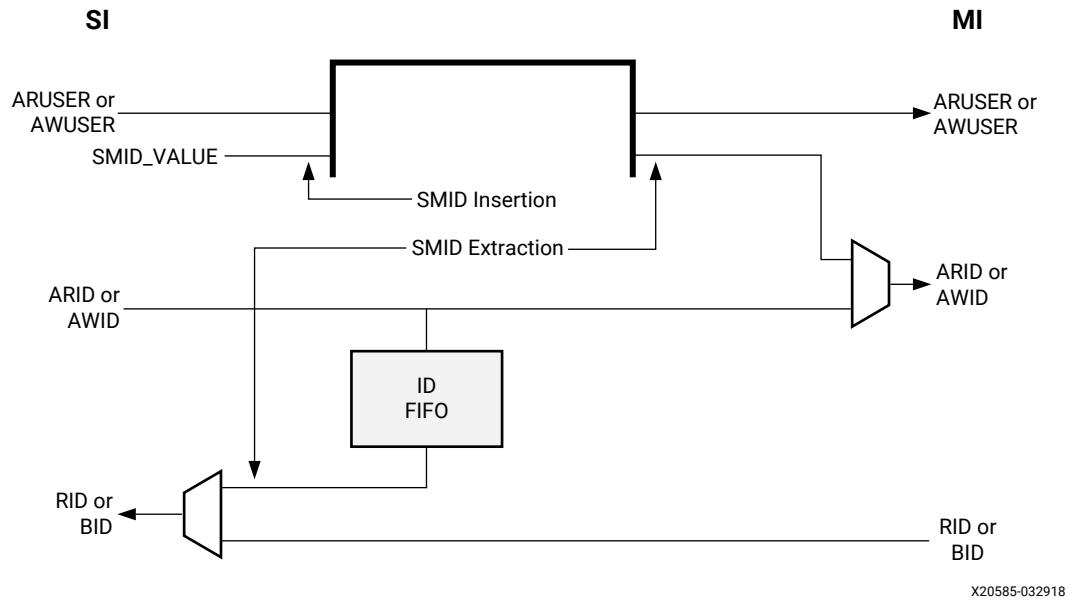
Parity

- Generates parity on Write Data and Read Data channel output (optional).
- Detect parity errors on Write and Read channel input (optional).

Operation

The functional block diagram of the core is shown in the following figure.

Figure 1: **SMID Functionality**



The SMID Functionality figure shows the SMID functionality of the core. When SMID_MODE=INSERT, the constant SMID_VALUE is inserted into the low-order position of the out-bound `aruser` or `awuser`, for each read or write command, to be transported across the SmartConnect network. When SMID_MODE=EXTRACT, the SMID value is recovered from the in-bound user signal and transferred onto the out-bound `arid` or `awid` to be presented to a PL-PS AXI-slave interface of a MPSoC block.

SMID for MPSoC SMMU

In Zynq® UltraScale+™ MPSoC, the processor SMMU performs address translation based on a System Management ID tag (SMID). See the *Zynq UltraScale+ Device Technical Reference Manual* (UG1085). The SMID identifies the master device that is accessing one of the slave devices in the processor block via the SMMU. SMID values are normally communicated among masters and slaves within the Processing System hard-block. AXI interfaces of the fabric IP do not natively communicate SMID information. However, SMID values can be transported across the fabric as sideband information as part of the AXI `awuser` or `aruser` signal. You can connect the AXI Sideband Formatter Utility IP core along AXI pathways in the fabric to insert and extract SMID values so that fabric masters can take advantage of SMMU address translation when accessing PS block slaves.

For each fabric master that accesses the PS block, you can insert an AXI Sideband Formatter Utility IP core between the master's AXI interface and the fabric interconnect. You then configure the IP to insert an SMID value for that master into the AXI `awuser` or `aruser` signal. You can connect another AXI Sideband Formatter IP core between the fabric interconnect and one of the AXI fabric slave interfaces of the Zynq UltraScale+ MPSoC Processing System core. You then configure this slave-side IP to extract the SMID value and pass it to the PS block in the format expected by the SMMU.

SMID Insertion

To set the SMID value for a fabric master, you connect its AXI master interface to the S_AXI interface of an AXI Sideband Formatter Utility IP core. Then connect the M_AXI interface of the IP to the fabric interconnect so that transactions can reach the PS block. You configure the IP to perform SMID insertion, specifying the field width and constant SMID value.

For each AW or AR transfer, the value of SMID is inserted into the low-order position of `awuser` and `aruser`; any in-bound value is left-shifted.

SMID Extraction

To retrieve the SMID value from an AXI transaction arriving at the PS block, you insert an AXI Sideband Formatter Utility IP core between the fabric interconnect and one of the AXI fabric slave interfaces of the PS block. You then configure the IP to perform SMID extraction, again specifying the width of the SMID field to extract.

It is also acceptable to directly connect an AXI Sideband Formatter Utility IP core configured for SMID extraction to the output of another AXI Sideband Formatter Utility IP core configured for SMID insertion, if no intervening AXI interconnect is needed. (These two functions cannot be performed by the same instance of the core.)

For each AW or AR transfer, the SMID field is extracted from the low-order position of `awuser` or `aruser` and the remainder (if any) is right-shifted and propagated.

The extracted SMID value is propagated as the out-bound `awid` or `arid` (and `wid` if AXI3), which is where the SMMU expects to find the SMID.

Any in-bound value of `awid` or `arid` (if S_ID_WIDTH>0) is pushed into a queue to be restored during the corresponding response transfer.

- During response transfers, received `rid` or `bid` are discarded and replaced by the saved `awid` and `arid` (if any).
- AW and AR command traffic gets single-threaded according to SMID (disregarding any `awid` or `arid`). Any transfer received while there is an outstanding transaction with a different SMID value gets stalled. This avoids any response reordering.

In AXI3, W-channel gets stalled until the corresponding AW is received so that extracted SMID can be propagated as `wid`.

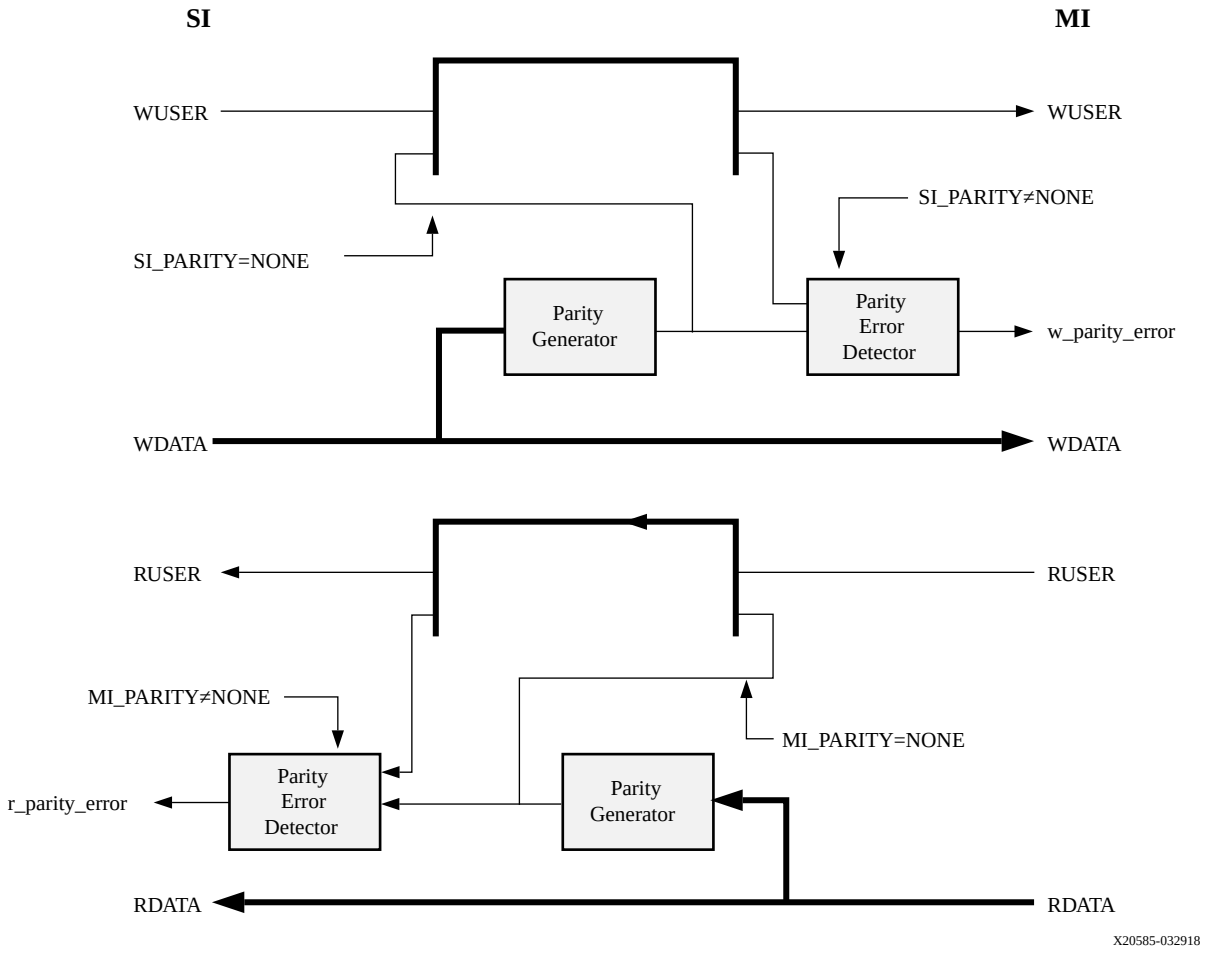
SMID Removal

When the AXI `awuser` and `aruser` signals are being used to carry other information in addition to SMID transport, the AXI Sideband Formatter Utility IP core restores the original value when the SMID is extracted at the PS block interface. But if the user signal information also needs to go to destinations other than the PS block, the core can also be used to restore the original information without processing the SMID. By inserting an AXI Sideband Formatter Utility IP core in front of any AXI slave endpoint and configuring it to perform SMID removal, the IP will strip the SMID and propagate the original info to the slave.

For each AW or AR transfer, the SMID field is extracted from the low-order position of `awuser` or `aruser` and the remainder (if any) is right-shifted and propagated.

Parity

Figure 2: **Parity Functionality**



The Parity Functionality figure shows the parity functionality of the core. When enabled, parity is either generated on the out-bound `wuser` or `ruser` signal or checked for errors against the in-bound data. The parameters SI_PARITY and MI_PARITY determine whether the core acts as a parity endpoint or whether it propagates parity information through while checking for errors.

AXI protocol does not define a dedicated parity signal to accompany its read and write data channels. If parity protection is wanted, the AXI `ruser` and `wuser` signals can transport one parity bit per byte of data. AXI masters can be designed to generate parity for their `wdata` output, and detect parity errors on their `rdata` inputs, based on parity bits transported on their `wuser` or `ruser` ports. Similarly, AXI slaves can generate parity for `rdata` outputs and detect

parity for wdata inputs. When designing with masters or slaves that do not generate/detect parity on their own, the AXI Sideband Formatter Utility IP core can generate and detect parity on their behalf. By inserting the AXI Sideband Formatter Utility IP core adjacent to the masters and/or slaves, you can protect data transmissions across the interconnect topology, including the various storage elements along the pathway.

Generating Parity

When deploying the AXI Sideband Formatter Utility IP core to perform parity operations, its configuration parameters designate whether parity information exists in the wuser/ruser signals on the S_AXI interface and on the M_AXI interface. When one of the interfaces indicates that it carries no parity and the opposite interface indicates that it does carry parity, then parity bits are generated for the out-bound data channel on the interface with parity enabled. For example, if parity is not present (disabled) on SI and enabled on MI, then parity is generated for the MI W-channel and inserted into the m_axi_wuser output signal, and vice-versa for read parity.

As data channel transfers propagate through the interconnect topology, information contained in the wuser or ruser signal, including parity, remain associated with the corresponding bytes of data at all times. For example, if the interconnect performs a data-width conversion along the AXI pathway, the parity bits associated with the data bytes in each data-beat of the burst, before and after the conversion, remain in the same data-beat as the associated data, and in the correct order. That is why the dimensions of the wuser and ruser signals are always expressed as an integer number of user-bits per byte of data.

For each W-channel or R-channel transfer, parity is generated per byte of data and transmitted in the lowest-order bit-per-byte position of the out-bound wuser or ruser. When generating parity, any in-bound user signal value is left-shifted 1 bit-per-byte (parity bits are interleaved among other byte-related payload). Parity is generated according to the polarity (ODD or EVEN) specified in the configuration parameters.

Parity is generated for all write data byte positions regardless of whether the corresponding wstrb bit is asserted, so that the result can never be misinterpreted as a parity violation.

Detecting Parity

When either of the SI or MI interfaces indicates that it carries parity (enabled), the AXI Sideband Formatter Utility IP core performs parity error detection on the in-bound data channel of each interface where parity is enabled, regardless of whether the opposite interface is also enabled for parity. Parity detection is performed based on the polarity (ODD or EVEN) selected for the interface where the data is received.

If parity is not also being propagated to the opposite interface, the parity bits are stripped and any remaining user-signal value is right-shifted 1 bit-per-byte.

If parity is also enabled on the opposite interface (where the corresponding data channel is out-bound), then in-bound parity information is propagated as-is, including any parity mismatches. If the polarity of the out-bound interface is the same as the in-bound interface (ODD-to-ODD or EVEN-to-EVEN), then parity information is propagated without inversion. If the out-bound polarity is opposite the in-bound interface, then in-bound parity info is inverted and propagated (including any parity mismatches). The AXI Sideband Formatter Utility IP core never generates parity info for in-bound data when the in-bound interface is enabled for parity; it just passes the in-bound parity info through, while checking for errors.

If any parity mismatches are detected in any byte lane of data, the AXI Sideband Formatter Utility IP core asserts an error condition during the valid/ready handshake completion cycle only (not sticky). The error condition is driven onto the `w_parity_error` or `r_parity_error` output signal during the same or subsequent cycle, depending on whether there is any pipelining configured for the core. Edge-triggered interrupts can be used to trap parity violations. Connecting to a counter-enable can be used to determine the number of data beats in which any parity violations occurred.

The assertion of `w_parity_error` or `r_parity_error` can optionally be pipelined to improve timing. Propagation of W-channel or R-channel payload between SI and MI interfaces remains combinatorial, regardless of error output pipelining.

Write parity detection is masked per deasserted bit of `wstrb`.

IMPORTANT: Data aggregation during width conversion in the interconnect may produce filler bytes of all-zero DATA and all-zero USER. For that reason, even parity is recommended when propagating across SmartConnect.

Parity Operating Mode

Table 1: **Parity Operating Mode**

SI_PARITY	MI_PARITY	Description	Change in WUSER bits-per-byte from SI to MI	Change in RUSER bits-per-byte from MI to SI
NONE	NONE	All parity functions disabled; Propagate any USER signals as-is.	+0	+0
NONE	{EVEN,ODD}	Generate write parity;Detect/strip read parity.	+1	-1
{EVEN,ODD}	NONE	Generate read parity; Detect/strip write parity.	-1	+1
{EVEN,ODD}	{EVEN,ODD}	Propagate write and read parity as-is;Detect any in-bound parity errors.	+0	+0

Table 1: **Parity Operating Mode** *(cont'd)*

SI_PARITY	MI_PARITY	Description	Change in WUSER bits-per-byte from SI to MI	Change in RUSER bits-per-byte from MI to SI
{EVEN,ODD}	{ODD,EVEN}	Propagate inverted write and read parity (including mismatches); Detect any in-bound parity errors.	+0	+0

Unsupported Features

The following features of the standard are not supported in the core:

- The IP core is not supported for AXI4-Lite.
- In some modes, payload propagation between the SI and MI is not pipelined. That includes combinatorially generating parity on the data channels on-the-fly. If any pipelining is required to close timing, register slices should be connected to the SI or MI as needed.

Licensing and Ordering

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

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

Product Specification

The following subsections describe the core specifications.

Standards

This core adheres to the AXI4 and AXI3 standards.

Performance

Not Applicable for this core.

Resource Use

Not Applicable for this core.

Port Descriptions

Table 2: **Signal Interfaces**

Interface	Signals	Dir	Width	Enablement	Description
s_axi	{ar,aw}addr	I	ADDR_WIDTH	always	Transaction Address
	{ar,aw,r,b,w}id	-	S_ID_WIDTH	S_ID_WIDTH>0	AXI ID
	{ar,aw,w,r,b}user	-	S_{AR,AW,W,R,B}USER_WIDTH	*USER_WIDTH>0	AXI user signal. May also carry SMID or parity information.
	{r,w}data	-	DATA_WIDTH	always	Data payload
	other AXI signals	-	-	-	as per AMBA® AXI4 specification
m_axi	{ar,aw}addr	O	ADDR_WIDTH	always	Transaction Address
	{ar,aw,r,b,w}id	-	M_ID_WIDTH	M_ID_WIDTH>0	AXI ID If SMID Extraction enabled, this value is replaced by the SMID constant and sent to the PS fabric interface.
	{ar,aw,w,r,b}user	-	M_{AR,AW,W,R,B}USER_WIDTH	*USER_WIDTH>0	AXI user signal. May also carry SMID or parity information.
	{r,w}data	-	DATA_WIDTH	always	Data payload
	other AXI signals	-	-	-	as per AMBA AXI4 spec

Table 2: **Signal Interfaces** *(cont'd)*

Interface	Signals	Dir	Width	Enablement	Description
(out-of-band)	w_parity_error	O	1	always	If SI_PARITY=={EVEN,ODD},asserted for 1 cycle while s_axi_wvalid and s_axi_wready if a write parity error has been detected in any byte lane during the current data transfer (not sticky). Error is masked by deasserted wstrb. Output signal may be delayed from detected error condition by one clock cycle per enabled bit of ENABLE_PIPELINING_PARITY. Error = for any byte lane i: (^s_axi_wdata[i*8 +: 8] ^ s_axi_wuser[i*S_WUSER_BITS_PER_BYTE] ^ (SI_PARITY==ODD)) & s_axi_wstrb[i]
	r_parity_error	O	1	always	If MI_PARITY=={EVEN,ODD}, asserted for 1 cycle while m_axi_rvalid & m_axi_rready if a read parity error has been detected in any byte lane during the current data transfer (not sticky). Output signal may be delayed from detected error condition by one clock cycle per enabled bit of ENABLE_PIPELINING_PARITY. Error = for any byte lane i: ^m_axi_rwdata[i*8 +: 8] ^ m_axi_ruser[i*M_RUSER_BITS_PER_BYTE] ^ (MI_PARITY==ODD)
	w_parity_error_injection	I	1	always	Force w_parity_error output high while s_axi_wvalid & s_axi_wready are asserted regardless of error detection. Output signal may be delayed from valid/ready handshake cycle by one clock cycle per enabled bit of ENABLE_PIPELINING_PARITY.
	r_parity_error_injection	I	1	always	Force r_parity_error output high while m_axi_rvalid & m_axi_rready are asserted regardless of error detection. Output signal may be delayed from valid/ready handshake cycle by one clock cycle per enabled bit of ENABLE_PIPELINING_PARITY.

Table 2: **Signal Interfaces** *(cont'd)*

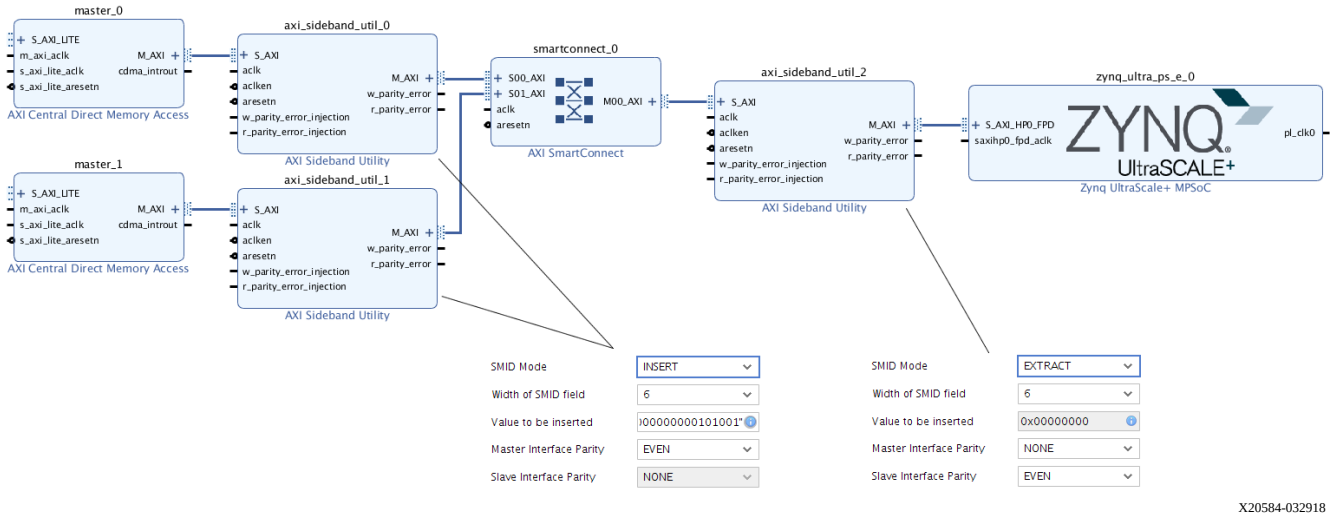
Interface	Signals	Dir	Width	Enablement	Description
aclk	aclk	I	1	always	Clock for internal state.
aclken	aclken	I	1	always	Clock enable for internal state; connection optional (default 1)
aresetn	aresetn	I	1	always	Active-Low reset for internal state; connection optional (default 1). This IP may propagate AXI handshake signals combinatorially and without gating handshake outputs with aresetn; in-bound handshakes asserted during reset may cause unpredictable output transfers.

Designing with the Core

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

Application Example

Figure 3: **Application Example**



The Application Example figure shows how the AXI Sideband Formatter Utility IP core is typically used in an IP Integrator design.

Fabric masters "master_0" and "master_1" access the memory controller in the MPSoC block. The SMMU in the block performs address translation depending on which master is accessing it. Sideband Formatter instances 0 and 1 are each configured to perform SMID Insertion, specifying a constant SMID value to distinguish each master. The SMID value gets inserted into the `aruser` and `awuser` signals for each command issued by each master, and propagate through the SmartConnect to reach the MPSoC. Instance "axi_sideband_util_2" performs SMID Extraction, retrieving the value from the user signal and transferring it to the `arid` or `awid` input to the MPSoC block.

In this example, data transfers between the two fabric masters and the MPSoC through the SmartConnect are also parity-protected. All three Sideband Formatter instances establish parity endpoints, enveloping the AXI pathways from instances 0 and 1 to instance 2. Parity for the write data channel is generated by instances 0 and 1, and is checked for parity violations at instance 2. Similarly, read data parity is generated by instance 2, and checked at each of instance 0 and 1. Typically the `w_parity_error` output of axi_sideband_util_2 and the `r_parity_error` outputs of instances 0 and 1 would either be connected to an interrupt controller or a ChipScope ILA (not shown).

Clocking

All I/O signals on the IP core are synchronized to the `aclk` input, except those that are propagated combinatorially.

Resets

The IP core requires one active-Low reset for all interfaces, `aresetn`. The reset is synchronous to `aclk`. AXI networks connected to the SI and MI interfaces should be reset concurrently with this IP.

Design Flow Steps

This section 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)
- *Vivado Design Suite User Guide: Designing with IP* (UG896)
- *Vivado Design Suite User Guide: Getting Started* (UG910)
- *Vivado Design Suite User Guide: Logic Simulation* (UG900)

Customizing and Generating the Core

This section includes information about using Xilinx® tools to customize and generate the core in the Vivado® Design Suite.

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl console.

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:

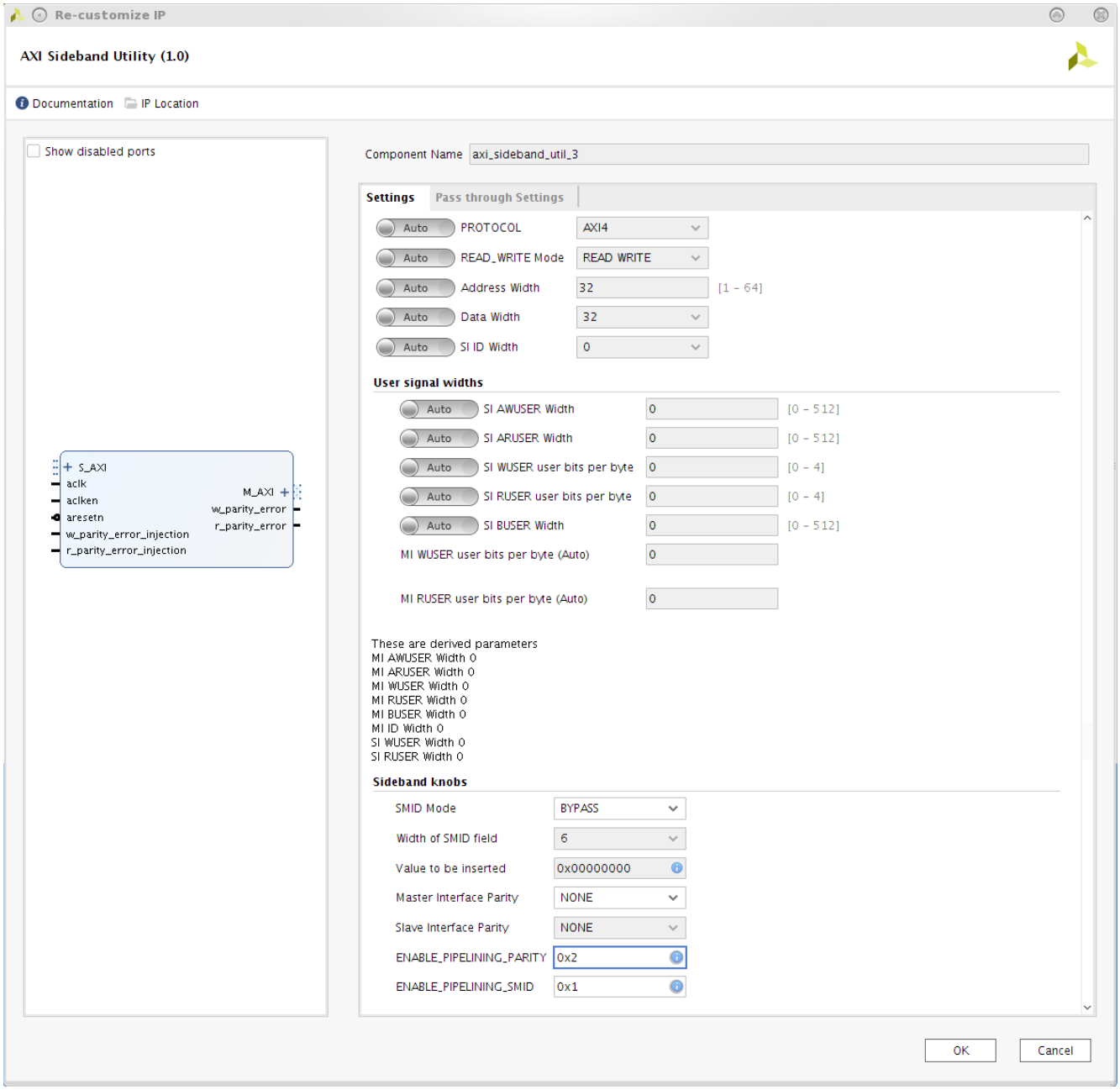
1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

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

Figures in this chapter are illustrations of the Vivado IDE. The layout depicted here might vary from the current version.

Configuration Tab 1

Figure 4: **Configuration Tab 1**



X20587-032918

User Parameters

The following table shows the relationship between the fields in the Vivado IDE and the user parameters (which can be viewed in the Tcl Console).

Table 3: **User Parameters**

Model Parameter	Format/Range	Default	Description
SMID_MODE	string={BYPASS,INSERT,EXTRACT,REMOVE}	BYPASS	BYPASS: No SMID handling. Any in-bound awuser/aruser info is passed through unchanged. INSERT: The value of SMID_VALUE is inserted into the low-order position of {aw,ar}user; any in-bound value is left-shifted. EXTRACT: The SMID field is extracted from the low-order position of {aw,ar}user and propagated as the out-bound {aw,ar}id (and wid if AXI3). REMOVE: The SMID field is extracted from the low-order position of {aw,ar}user and discarded; the remainder (if any) is right-shifted and propagated
SMID_VALUE	bitstring(width=SMID_WIDTH)	6'b0	The value to be inserted into the low-order position of {aw,ar}user.
SMID_WIDTH	integer={0..32}	6	Width of SMID field.

Table 3: **User Parameters** *(cont'd)*

Model Parameter	Format/Range	Default	Description
MI_PARITY	string={NONE, ODD, EVEN}	NONE	If EVEN or ODD, parity is generated or propagated per byte of write data and is output in the lowest-order bit-per-byte position of the out-bound m_axi_wuser. If SI_PARITY==NONE, parity is generated and inserted, any in-bound value from s_axi_wuser is left-shifted 1 bit-per-byte. If SI_PARITY is the same as MI_PARITY (EVEN or ODD), in-bound parity is propagated as-is (including any parity mismatches). If SI_PARITY is the opposite polarity to MI_PARITY, in-bound parity is inverted and propagated (including any parity mismatches). If EVEN or ODD, parity is expected in the lowest-order bit-per-byte position of the in-bound m_axi_ruser and compared against the corresponding read data byte for parity violation errors. If NONE, any parity received on the SI W-channel is stripped from the lowest-order bit-per-byte position of the out-bound m_axi_wuser, and any remaining value is right-shifted 1 bit-per-byte. If NONE, no parity is expected in the in-bound R-channel.

Table 3: **User Parameters** *(cont'd)*

Model Parameter	Format/Range	Default	Description
SI_PARITY	string={NONE, ODD, EVEN}	NONE	If EVEN or ODD, parity is generated or propagated per byte of read data and is output in the lowest-order bit-per-byte position of the out-bound s_axi_ruser. If MI_PARITY==NONE, parity is generated and inserted, any in-bound value from m_axi_rwuser is left-shifted 1 bit-per-byte. If MI_PARITY is the same as SI_PARITY (EVEN or ODD), in-bound parity is propagated as-is (including any parity mismatches). If MI_PARITY is the opposite polarity to SI_PARITY, in-bound parity is inverted and propagated (including any parity mismatches). If EVEN or ODD, parity is expected in the lowest-order bit-per-byte position of the in-bound s_axi_wuser and compared against the corresponding write data byte for parity violation errors. If NONE, any parity received on the MI R-channel is stripped from the lowest-order bit-per-byte position of the out-bound s_axi_ruser, and any remaining value is right-shifted 1 bit-per-byte. If NONE, no parity is expected in the in-bound W-channel. If SI_PARITY==NONE and MI_PARITY==NONE, then all parity functions are disabled and any {w,r}user signals are propagated as-is.

Table 3: **User Parameters** *(cont'd)*

Model Parameter	Format/Range	Default	Description
S_{AW,AR,B}USER_WIDTH	0<=integer<=512	0	Width of s_axi_{ar,aw,b}user.
S_{R,W}USER_BITS_PER_BYTE	0<=integer<=4	0	Bits-per-byte ratio of s_axi_{w,r}user.
ADDR_WIDTH	1<=integer<= 64	32	Width of *_axi_{ar,aw}addr.
S_ID_WIDTH	0<=integer<= 32	0	Width of s_axi_{ar,aw,w,r,b}id.
DATA_WIDTH	integer=2**{5..10}	32	Width of *_axi_{r,w}data in bits.
PROTOCOL	string={AXI4,AXI3}	AXI4	Protocol of SI and MI interfaces
READ_WRITE_MODE	string={READ_WRITE,READ_ONLY,WRITE_ONLY}	READ_WRITE	Channel enablement
ENABLE_PIPELINING _PARITY	Bitstring, width=4 bits	"0010"	Adds 1 cycle latency between parity error detection and error output for each enabled bit (0-4 latency cycles). Adding pipelining can help timing closure. Bit 0 (LSB): Register the data input from the AXI interface (for error detection only). Bit 1: Pipeline the intermediate parity results for each byte of data (enabled by default). Bit 2: Insert a pipeline stage mid-way between byte-wise parity results and the error output. Bit 3: Register the error output.
ENABLE_PIPELINING_SMID	Bitstring, width=1 bit	"1"	When SMID_MODE=EXTRACT, incur 1 cycle of latency for the propagation from SI to MI for the AR and AW channels to improve timing (enabled by default).

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896).

Constraining the Core

Required Constraints

This section is not applicable for this IP core.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This section is not applicable for this IP core.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

Simulation

For comprehensive information about Vivado® simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900).

Synthesis and Implementation

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

Appendix A

Upgrading

This appendix is not applicable for the first release of the core.

Appendix B

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

Finding Help on Xilinx.com

To help in the design and debug process when using the core, the 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 core. 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.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use keywords such as:

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the Core

AR 70852

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.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

Appendix C

Additional Resources and Legal Notices

Xilinx Resources

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

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help** → **Documentation and Tutorials**.
- On Windows, select **Start** → **All Programs** → **Xilinx Design Tools** → **DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

***Note*:** For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this product guide:

1. AMBA® AXI and ACE Protocol Specification (ARM IHI0022E)
2. *AXI Protocol Checker LogiCORE IP Product Guide* (PG101)
3. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
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. *ISE to Vivado Design Suite Migration Guide* (UG911)
8. *Vivado Design Suite User Guide: Implementation* (UG904)
9. *SmartConnect LogiCORE IP Product Guide* (PG247)

Training Resources

1. Vivado Design Suite Hands-on Introductory Workshop
2. Vivado Design Suite Tool Flow

Revision Table

The following table shows the revision history for this document.

04/04/2018 v1.0
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 https://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 https://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.

Copyright