CHAIN® Network Adapter User Guide

(CHAINworks 2.1.1)

License

Disclaimer

Table of Contents

[THIS PAGE INTENTIONALLY LEFT BLANK]

Adapter Basics: Connecting Hardware IP to the Silistix Network-on-Chip (NoC)

What is an Adapter?

A Silistix adapter connects IP blocks in the system, based on one of the industry-standard interfaces shown in Table 1, to the Silistix CHAIN Network-on-Chip (NoC) architecture. The Silistix adapter provides all the necessary data protocol conversion and formatting.

Table 1: **Silistix NoC Adapters (May 2008)**

IP Standard	Initiator	Target
AMBA APB	N/A	✓
AMBA AHB	✓	✓
AMBA AXI	✓	✓
OCP	✓	✓

Each Silistix adapter is clocked using the same local or regional clock source that drives the attached hardware IP block. However, once converted, all traffic over the Silistix Network-on-Chip is self-timed and asynchronous.

Adapters are available as either an initiator or a target, per IP standard listed in Table 1. AMBA APB, however, is only defined as a target.

Initiator Adapters

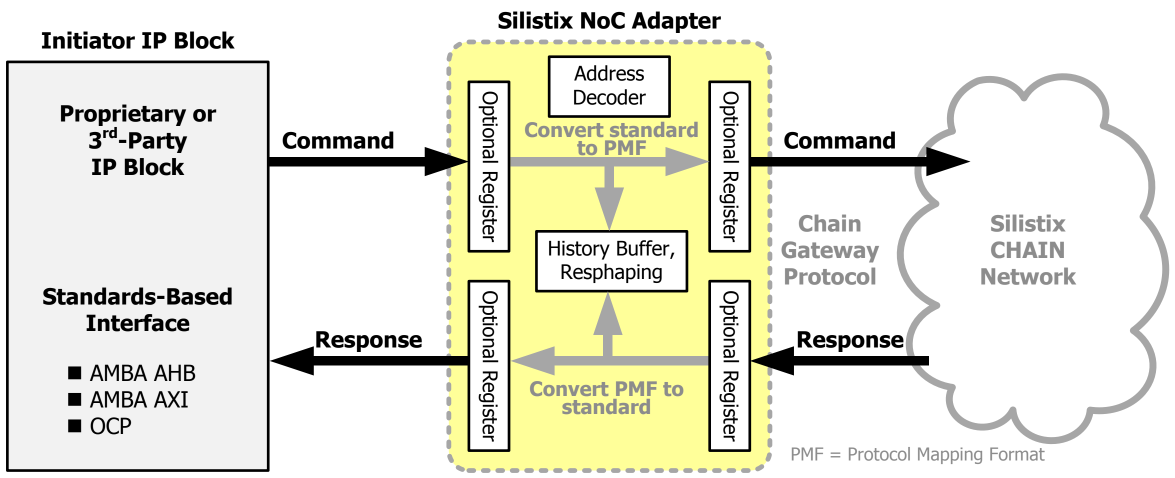
Figure 1 presents a top-level diagram of how an initiator IP block attaches to the Silistix NoC using a Silistix network adapter. When the IP block wishes to initiate a transaction, it generates any command, address, and data as might be required by the native IP interface protocol. The Silistix initiator adapter converts this request into an internal form called Protocol Mapping Format (PMF). The PMF format enables nearly transparent transactions between different IP interface standards. These PMF packets then tunnel through the Silistix Chain Gateway Protocol (CGP) network.

The initiator adapter also decodes the address for any transaction so that packets are correctly routed to their proper destination.

The addressed target at the other end of the network responds to commands sent by the initiator. Its response returns over the CGP network as PMF packets. The initiator adapter reconverts these PMF packets into the native protocol understood by the initiator IP block.

In some applications, the initiator adapter also reshapes data packets. For example, if a 64-bit initiator communicates to an 8-bit target, then the adapter converts each 64-bit transaction into eight byte-wide transactions.

Figure 1: **Example Initiator Adapter**



Because both the initiator and target IP blocks communicate through the Silistix network, they can have the same or completely different native formats, courtesy of the Silistix Protocol Mapping Format (PMF). For example, an OCP-based initiator can communicate with an APB-based target.

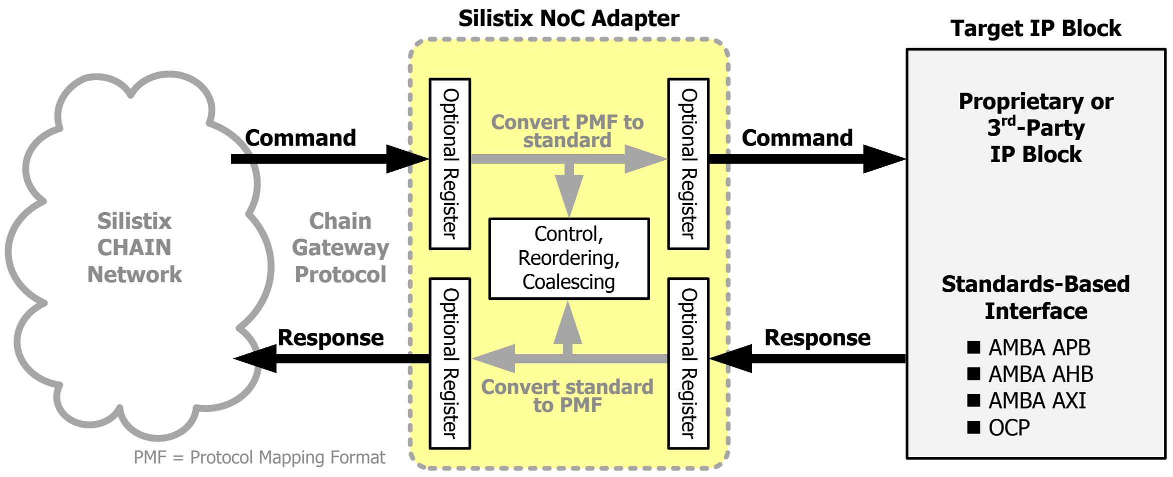
The History Buffer tracks and manages any potential outstanding requests. Some of the protocol adapters, such as AXI, allow there to be multiple outstanding transactions on the network.

Target

Figure 2 presents a top-level diagram of how a target IP block attaches to the Silistix NoC using a Silistix network adapter. The address decoding for the target, as well as the any transaction arbitration, is seamlessly handled within the Silistix network.

The addressed target responds to the command sent by the transaction initiator. The adapter converts the command, sent in PMF, into the native protocol supported by the IP block. The response from the target, converted back into PMF packets, is sent back to the initiator over the Silistix CGP network. The initiator adapter at the other end of the network converts these PMF packets into the native protocol understood by the initiator IP block.

Figure 2: **Example Target Adapter**



In some applications, the target adapter also performs data reordering, trimming, or coalescing functions.

Declaring Adapter-specific Options in CSL

Generally, there is no need to specify an adapter-specific option unless the application requires a value different than the default. However, there are also mandatory options for some protocol adapters. For example, the AXI adapter requires a setting to control the width of the AXI ID signals. As shown in Figure 3, these adapter-specific options can be included with the other statements in the CSL source file.

Figure 3: **Adapter-specific Options Declared in CSL Source File**

```
clock_domain cpu_domain (400 MHz) {
   CPU_endpoint {
      protocol = “AXI”;

      initiator CPU_initiator {
         address = 32 bits;
         data = 32 bits;
         address_map = CPU_address;
         write_response = 15 ns;
         read_response = 30 ns;
         // adapter-specific options
         axi_id_bits = 1;
      } //end CPU_initiator

      target CPU_mailbox {
         address = 4 bits;
         data = 16 bits;
         burstsize = 16 bytes;
         address_range = COMM_address.CPU_mailbox;
         // adapter-specific options
         axi_id_bits = 1;
         axi_write_interleave_depth = 1;
      } // end CPU_mailbox target

   } // end CPU_endpoint
} // end clock_domain
```

Related Documents

Please refer to the following documents for additional information on how to use the CSL language to describe a system and how to process the design using the Silistix CHAINarchitect software.

- ***Building and Analyzing On-Chip Networks using CHAINarchitect***
- ***Describing a System Using Connection Specification Language (CSL™)***
- ***CSL Language Reference Manual***

[THIS PAGE INTENTIONALLY LEFT BLANK]

AMBA Peripheral Bus (APB) Adapter

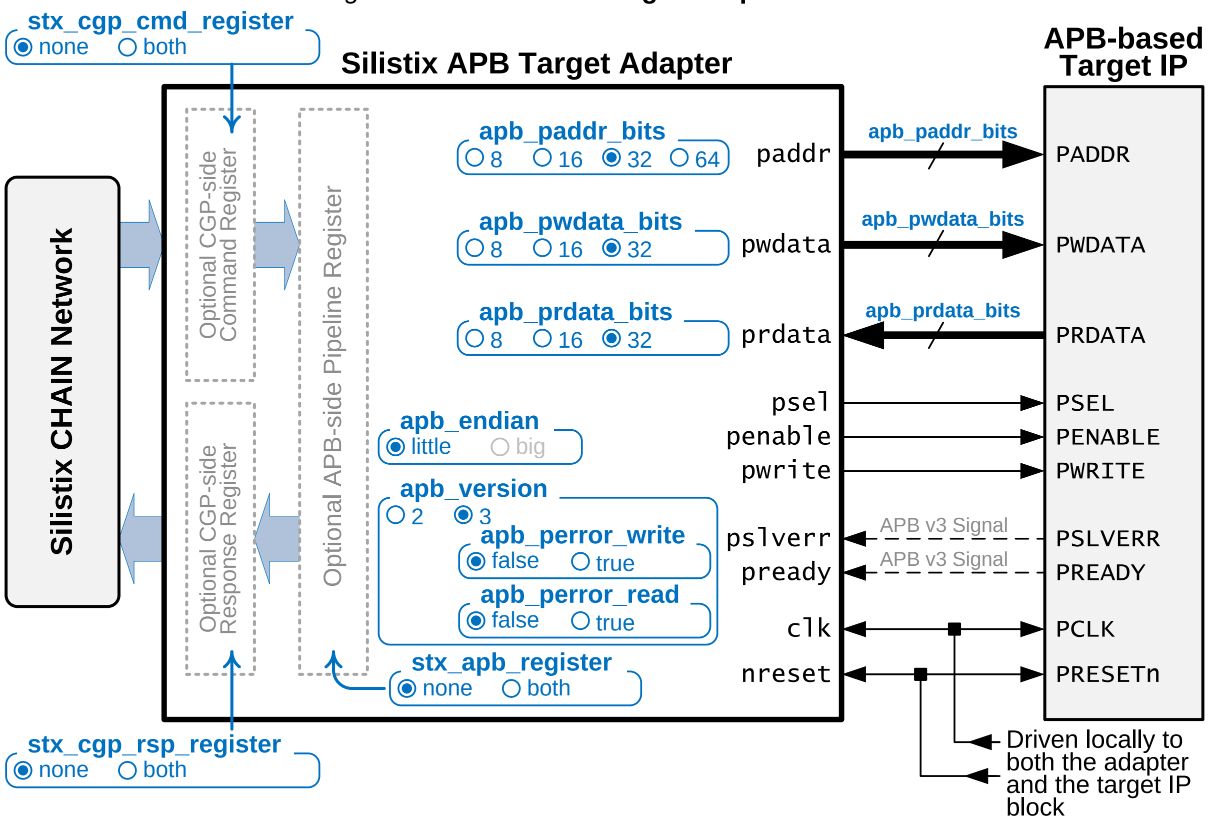
The Silistix APB adapter, shown in Figure 4, is designed according to the ARM AMBA Peripheral Bus standard. The AMBA Peripheral Bus (APB) is only supported as a target.

The APB adapter supports both version 2 and 3 of the protocol. Version 3 of the protocol adds the **`PREADY`** signal and the **`PSLVERR`** signal.

Support for multiported APB targets is provided in the adapter.

Customize the APB target adapter as required for the specific application by setting the attributes described below and summarized in Table 2.

Figure 4: **Silistix APB Target Adapter and Attributes**



The **`PCLK`** clock signal to the target IP block also drives the clocked logic within the APB target adapter. Similarly, the target adapter is reset by the active-Low **`PRESETn`** signal.

General Attributes

The following attributes are those that apply to all APB target applications. The name of each attribute is listed along with the allowable values. The default setting for the attribute appears in **<u>Bold Underline</u>**.

Protocol Version

The **apb_version** attribute defines which version of the interface is generated. The primary difference is that version 3 adds the **PREADY** signal and the **PSLVERR** signal.

```
apb_version = { 2 | 3 }
```

When using version 3, all see the APB Version 3 Options section on page 12.

Endian

The current APB target adapter only supports little endian data ordering.

```
apb_endian = { little }
```

Address Bus Width

The **apb_paddr_bits** attribute defines the width of the APB address bus, measured in bits.

```
apb_paddr_bits = { 8 | 16 | 32 | 64 }
```

Data Bus Width

The APB interface has separate read and write data busses. The bit width of each data bus is independently defined using the **apb_prdata_bits** and **apb_pwdata_bits** attributes.

Read Data Bus Width

```
apb_prdata_bits = { 8 | 16 | 32 }
```

Write Data Bus Width

```
apb_pwdata_bits = { 8 | 16 | 32 }
```

APB Version 3 Options

When using version 3 of the APB interface (**apb_version = 3**), then a set of attributes defines if the hardware IP block attached to the target adapter can generate an error response on the **PSLVERR** signal in response to a read or write operation. By default, both read and write operations do not report errors. By default, both read and write operations do not support error handling or reporting. If the attached target handles errors, set the write and read attributes `true`, as appropriate for the application.

Write Error Control

```
apb_perror_write = { false | true }
```

Read Error Control

```
apb_perror_read = { false | true }
```

Both attributes are automatically set to false if using version 2 of the APB interface (**apb_version = 2**).

Optional Pipeline Register Control

As shown in Figure 4, pipeline registers are optionally available for both the CGP-side and the APB-side adapter interfaces. If inserted, then the register uses the same clock source as the APB target IP block. In general, the default setting of `none` is preferred.

APB-side Interface

By default, there are no additional pipelining registers on the APB interface signals. Set **`stx_apb_register = both`** to add a pipeline register to the inputs and outputs of the APB-side interface. Inserting these pipeline registers adds levels of latency.

```
stx_apb_register = { none | both }
```

CGP-side Interface

The CGP-side interface has both a command path and a response path. The command path includes address, command, or write operations from the initiator to the selected target. The response path includes any read or status responses from the selected target back to the initiator.

Command Path

By default, there is no additional pipelining registers on the CGP command interface signals. Set **`stx_cgp_cmd_register = both`** to add a pipeline register to the inputs and outputs of the CGP-side command interface. Inserting these pipeline registers adds levels of latency.

```
stx_cgp_cmd_register = { none | both }
```

Response Path

By default, there is no additional pipelining registers on the CGP response interface signals. Set **`stx_cgp_rsp_register = both`** to add a pipeline register to the inputs and outputs of the CGP-side response interface. Inserting these pipeline registers adds levels of latency.

```
stx_cgp_rsp_register = { none | both }
```

Advanced Options

Target Data Width Wider than Initiator

The following attributes control the adapter behavior when the data path width of the initiator is narrower than that of the APB target. The opposite case, where the initiator data path is wider than the target, is already handled by the initiator.

The target and initiator data values are aligned to the least-significant bit in the data word.

Write Data from Narrower Initiator to Wider Target

The **`stx_apb_narrow_write`** and **`stx_apb_write_burst_concat`** attributes control the adapter behavior when an initiator with a narrower data width than the APB target IP writes data to the target. The **`stx_apb_write_burst_concat`** attribute defines whether or not the target adapter will attempt to combine beats of narrower burst into fewer beats of a wider burst. The **`stx_apb_narrow_write`** attribute describes the behavior of the adapter when the burst is not of the correct width. Its options are either to zero pad the write data or to fail and return an error response. If the attribute **`stx_apb_write_burst_concat = true`**, then the **`stx_apb_narrow_write`** attribute defines the adapter's behavior when it is unable to combine the beats of the burst into a wider transfer, perhaps because the burst is too short.

The `stx_apb_write_burst_concat` attribute defines whether a burst transfer from a narrower initiator is coalesced into one or more wider words for the wider APB target. There is additional area overhead when this attribute is set to `true`.

```
stx_apb_write_burst_concat = { false | true }
```

The `stx_apb_narrow_write` attribute defines whether such an operation should cause the target to generate an error response on such an attempted write operation (the default), or if the more-significant data bits are padded with zeroes. This attribute prevents a mis-sized write operation.

```
stx_apb_narrow_write = { fail | zeropad }
```

Read Data from Wider Target back to Narrower Initiator

The adapter returns the correct bytes from the wider APB target, based on the address, back to the narrower initiator. Any excess, unaddressed data from the resulting target access is discarded because it is now requested by the initiator. This type of transaction is supported by default. If this attribute is set to `fail`, then the target generates an error response when a narrow read is attempted, preventing a mis-sized read operation.

```
stx_apb_narrow_read = { fail | supported }
```

Multiported Endpoint Multiplexer/Demultiplexer Control

In a multiported application, a single adapter connects multiple APB targets. Read data from each target sub-port must be multiplexed together before returning to the initiator. Optionally, write data can be demultiplexed and presented to each target sub-port.

By default, these multiplexer and demulitiplexer functions are not implemented within the APB target adapter. However, the following options enable or disable these automatic functions in those applications that require the multiplexers and demultiplexer functions.

Initiator Multiport Read

When using a multi-ported APB interface, the `stx_apb_bridge_multiport_read` attribute defines whether the read multiplexer is automatically implemented within the adapter or manually defined within the target IP source code (the default).

```
stx_apb_bridge_multiport_read = { false | true }
```

Initiator Multiport Write

When using a multi-ported APB interface, the `stx_apb_bridge_multiport_write` attribute defines whether there is a signal write output port or multiple demultiplexed write output ports (the default).

```
stx_apb_bridge_multiport_write = { false | true  }
```

Table 2: **AMBA Peripheral Bus (APB) Adapter Attributes Summary**

Attribute Name	Default	Allowable Values	Description
General Attributes			
apb_version	`3`	`2`, `3`	Protocol version
apb_endian	`little`	`little`	Endian
apb_paddr_bits	`32`	`8`, `16`, `32`, `64`	Address bus bit width
apb_prdata_bits	`32`	`8`, `16`, `32`	Read data bus bit width
apb_pwdata_bits	`32`	`8`, `16`, `32`	Write data but bit width
APB Version 3 Options			
apb_perror_read	No default. Must specify	`true`, `false`	Controls whether read errors supported on **`PSLVERR`** signal
apb_perror_write	No default. Must specify	`true`, `false`	Controls whether write errors supported on **`PSLVERR`** signal
Optional Pipeline Register Control			
stx_apb_register	`none`	`none`, `both`	Optional ABP-side pipeline register
stx_cgp_cmd_register	`none`	`none`, `both`	Optional CGP-side command pipeline register.
stx_cgp_rsp_register	`none`	`none`, `both`	Optional CGP-side response pipeline register.
Target Data Width Wider than Initiator			
stx_apb_narrow_write	`fail`	`fail`, `zeropad`	Behavior during a narrow write operation.
stx_apb_write_burst_concat	`false`	`true`, `false`	Coalesce narrow initiator writes to wider target
stx_apb_narrow_read	`supported`	`fail`, `supported`	Behavior during a narrow read operation.
Multiported Endpoint Multiplexer/Demultiplexer Control			
stx_apb_bridge_multiport_read	`false`	`true`, `false`	Controls whether read multiplexer is in the adapter or the IP block.
stx_apb_bridge_multiport_write	`false`	`true`, `false`	Controls whether write demultiplexer is in the adapter or the IP block.

More Information

The Silistix APB adapters were created according the APB 2.0 and APB 3.0 specifications available under license from ARM.

- **AMBA 3 Specification**
  www.arm.com/products/solutions/axi_spec.html
- **AMBA 2 Specification**
  www.arm.com/products/solutions/AMBA_Spec.html

[THIS PAGE INTENTIONALLY LEFT BLANK]

AMBA High-Speed Bus (AHB) Adapter

The Silistix AHB adapters are designed according to the ARM AMBA AHB standard.

The AHB adapters support both the full and lite versions of the specification.

Customize the AHB adapters as required for the specific application by setting the attributes described below and summarized in Table 6.

Both the AHB initiator and target adapters have the identical level of support. However, there are minor differences between the initiator and target. The initiator adapter, shown in Figure 5, has an additional attribute (**abh_hlock**) and the target adapter, shown in Figure 6, has an additional signal (**HSEL**). Similarly, when the attribute **ahb_version = full**, then the initiator adapter also includes the signals designed for a multi-master interface, including the **HBURSTREQ** and **HGRANT** signals. The target adapter does not have these signals.

The **HCLK** clock signal to the target IP block also drives the clocked logic within the AHB adapter. Similarly, the target adapter is reset by the active-Low **HRESETn** signal.

General Attributes

The following attributes apply to all AHB adapter applications. The name of each attribute is listed along with the allowable values. The default setting for the attribute appears in **Bold Underline**.

Protocol Version

The **ahb_version** attribute defines which version of the interface is generated. The **full** version of the initiator adapter includes additional arbiter signals. The **lite** version of the initiator adapter is designed for a single-master system; the initiator arbiter signals are removed and the **HRESP** signal is reduced to a single bit for both the initiator and target adapters as shown in Table 3.

ahb_version = { **full** | lite }

Table 3: **AXI Full vs. Lite Signal Differences**

	Full	Lite
Initiator	HRESP[1:0] HBUSREQ HLOCK HGRANT	HRESP
Target	HRESP[1:0] HMASTER[3:0] HSPLIT[15:0]	HRESP

Endian

The current AHB adapter only supports little endian data ordering.

```
ahb_endian = { little }
```

Address Bus Width

In the current implementation of the adapter, the AHB address bus, **HADDR**, is always 32 bits wide.

Data Bus Width

The AHB interface has separate read and write data busses. The bit width of each data bus is independently defined by the `ahb_hrdata_bits` and `ahb_hwdata_bits` attributes.

Read Data Bus Width

```
ahb_hrdata_bits = { 8 | 16 | 32 | 64 }
```

Write Data Bus Width

```
ahb_hwdata_bits = { 8 | 16 | 32 | 64 }
```

Figure 5: **Silistix AHB Initiator Adapter and Attributes**

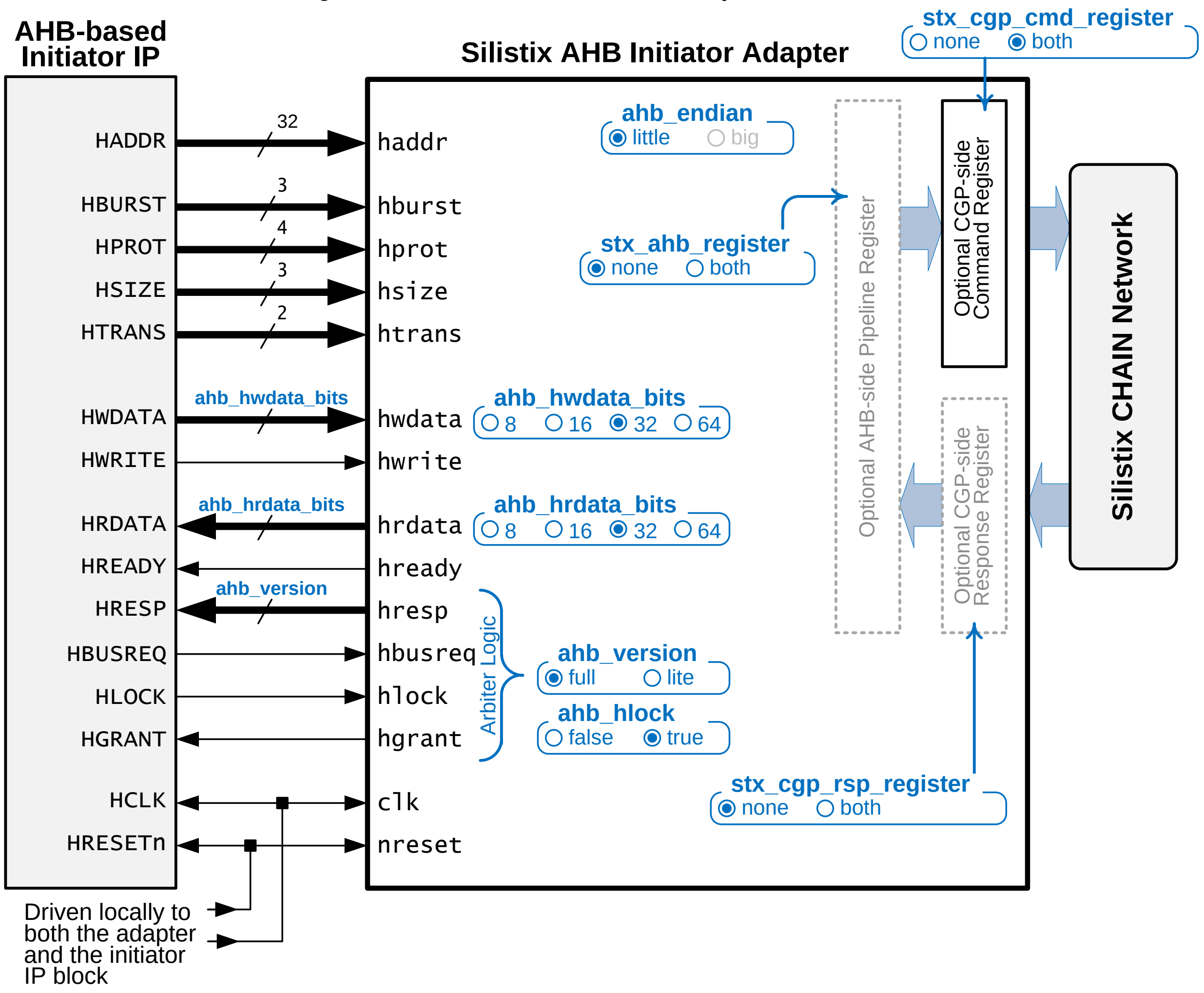
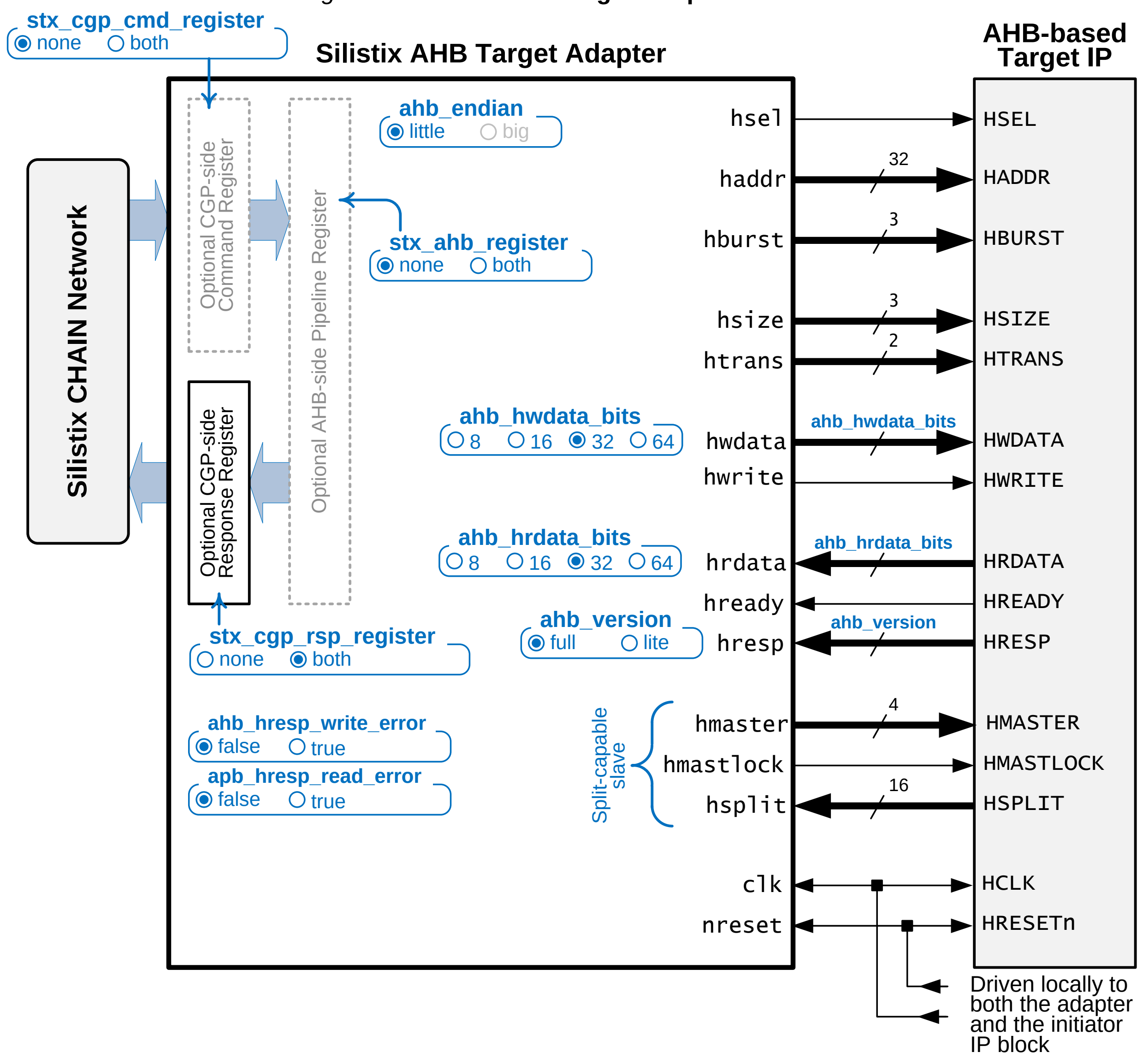


Figure 6: **Silistix AHB Target Adapter and Attributes**



Initiator-Only Attributes

The following attributes apply only to the AHB initiator adapter.

HLOCK Support

The AHB initiator adapter supports locked transfers by default. To disable locked transfers, set the attribute **ahb_hlock = false**.

```
ahb_hlock = { false | true }
```

The **abh_hlock** attribute controls different adapter signals, depending on the setting of the **ahb_version** attribute, as listed in Table 4.

Table 4: **Signal Affected by ahb_hlock Attribute**

ahb_version Attribute	Signal Controlled by ahb_hlock
full	HLOCK
lite	HMASTLOCK

Error Handling and Reporting

A set of attributes, **ahb_hresp_write_error** and **ahb_hresp_read_error**, define whether the hardware IP block attached the adapter handles errors. For a target adapter, these attributes indicate if the attached target endpoint (IP) generates an error response on the **HRESP** signal in response to a read or write operation. For an initiator adapter, these attributes indicate if the attached initiator endpoint (IP) understands errors reported on the **HRESP** signal. By default, both read and write operations do not support error handling or reporting. If the attached target or initiator handles errors, set the write and read attributes true, as appropriate for the application.

Write Error Control

```
ahb_hresp_write_error = { false | true }
```

Read Error Control

```
ahb_hresp_read_error = { false | true }
```

Optional Pipeline Register Control

As shown in Figure 4, pipeline registers are optionally available for both the CGP-side and the AHB-side adapter interfaces. If inserted, then the register uses the same clock source as the AHB IP block.

AHB-side Interface

By default, there is no additional pipelining registers on the AHB interface signals. Set **stx_ahb_register = both** to add a pipeline register to the inputs and outputs of the AHB-side interface. Inserting these pipeline registers adds levels of latency.

```
stx_ahb_register = { none | both }
```

CGP-side Interface

The CGP-side interface has both a command path and a response path. The command path includes address, command, or write operations from the initiator to the selected target. The response path includes any read or status responses from the selected target back to the initiator.

The CGP interface provides optional pipeline registers for both the command and response paths. Each register is individually controlled by an attribute as shown in Table 5. Setting the attribute to **both** inserts the pipeline registers to the associated signal group. Inserting these pipeline registers adds levels of latency.

To remove the pipeline registers, set the associated attribute to **none**.

Table 5: **CCP-side Pipeline Attributes**

CGP Path	Pipeline Register Control	Initiator Default Value	Target Default Value
Command	stx_cgp_cmd_register	both	none
Response	stx_cgp_rsp_register	none	both

Table 6: **AMBA High-speed Bus (AHB) Adapter Attributes Summary**

Attribute Name	Default	Allowable Values	Description
General Attributes			
ahb_version	`full`	`full`, `lite`	Protocol version
ahb_endian	`little`	`little`	Endian
ahb_haddr_bits	`32`	`32`	Address bus bit width
ahb_hrdata_bits	`32`	`8`, `16`, `32`, `64`	Read data bus bit width
ahb_hwdata_bits	`32`	`8`, `16`, `32`, `64`	Write data but bit width
Initiator-Only Attributes			
ahb_hlock	`true`	`true`, `false`	Allow locked transfers
Error Handling and Reporting			
ahb_hresp_write_error	`false`	`true`, `false`	Controls whether write errors supported on **`HRESP`** signal
ahb_hresp_read_error	`false`	`true`, `false`	Controls whether read errors are supported on **`HRESP`** signal
Optional Pipeline Register Control			
stx_ahb_register	`none`	`none`, `both`	Optional AHB-side pipeline register
stx_cgp_cmd_register	See Table 5	`none`, `both`	Optional CGP-side command pipeline register.
stx_cgp_rsp_register		`none`, `both`	Optional CGP-side response pipeline register.

More Information

The Silistix AHB adapters were created according the AHB specifications available under license from ARM.

- **AMBA 3 Specification**
  www.arm.com/products/solutions/axi_spec.html
- **AMBA 2 Specification**
  www.arm.com/products/solutions/AMBA_Spec.html

[THIS PAGE INTENTIONALLY LEFT BLANK]

Silistix®

AMBA Advanced eXtensible Interface (AXI) Adapter

The Silistix AXI adapters are designed according to the ARM AMBA AXI standard.

Customize the AXI adapters as required for the specific application by setting the attributes described below and summarized in Table 6.

Both the AXI initiator adapter, Figure 7, and the target adapter, Figure 8, have an identical level of support.

General Attributes

The following attributes apply to all AXI adapter applications. The name of each attribute is listed along with the allowable values. The default setting for the attribute appears in **Bold Underline**.

Endian

The current AXI adapter only supports little endian data ordering.

```
axi_endian = { little }
```

Address Bus Width

In the current implementation of the adapter, the AXI address busses, **AWADDR** and **ARADDR**, are always 32 bits wide.

Data Bus Width

The AXI interface has separate read and write data busses. The bit width of each data bus is independently defined by the **axi_rdata_bits** and **axi_wdata_bits** attributes.

Read Data Bus Width

```
axi_rdata_bits = { 8 | 16 | 32 | 64}
```

Write Data Bus Width

```
axi_wdata_bits = { 8 | 16 | 32 | 64}
```

ID Bit Width

The AXI adapter optionally presents or accepts identification tags for each group of signals. The width of all the ID signals (**AWID**, **WID**, **BID**, and **RID**) is controlled by a single attribute, **axi_id_bits**. There is **no default** for this attribute so this **mandatory value** must be specified.

```
axi_id_bits = { 1 | 2 | 4 }
```

Figure 7: **Silistix AXI Initiator Adapter and Attributes**

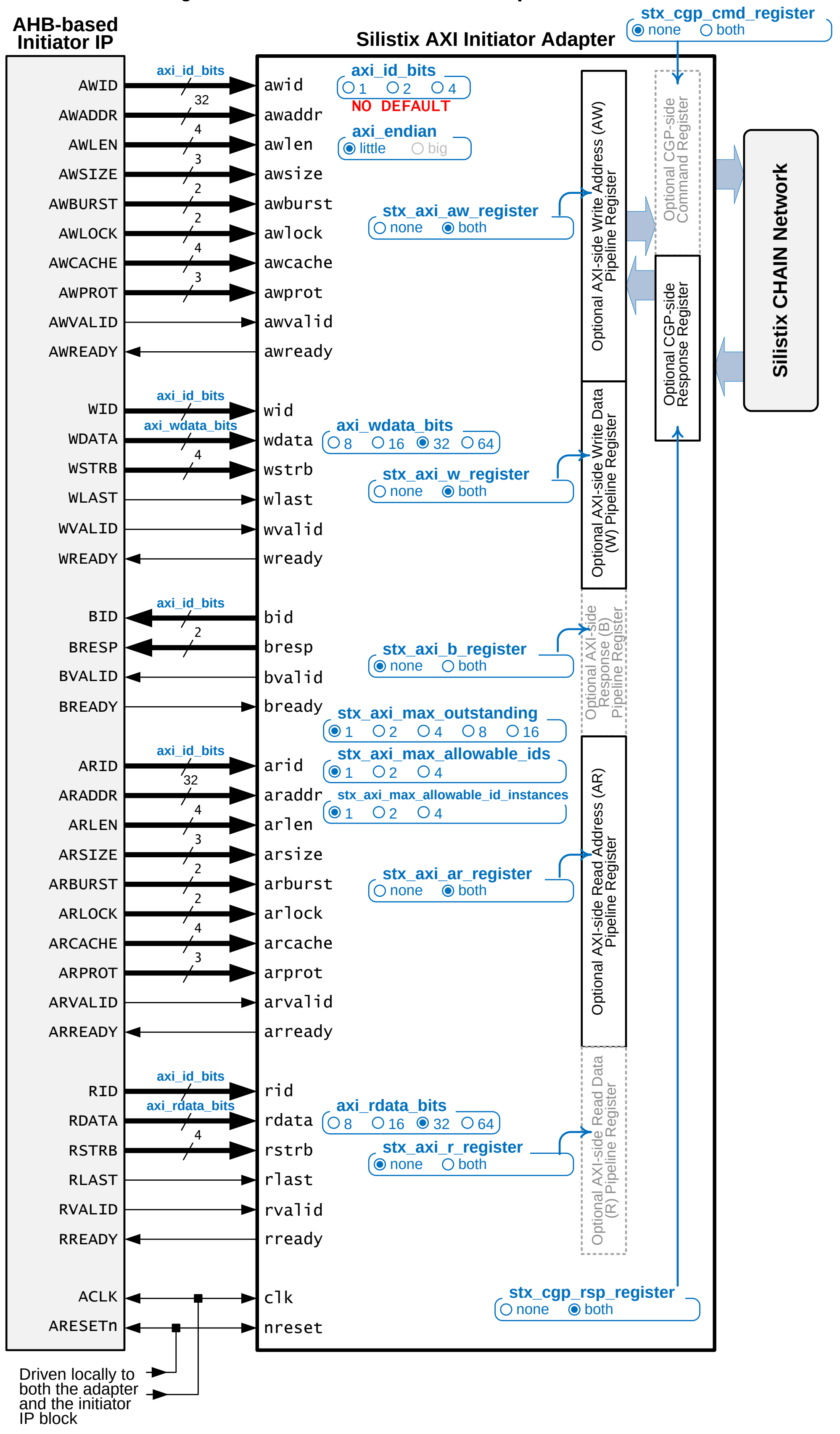
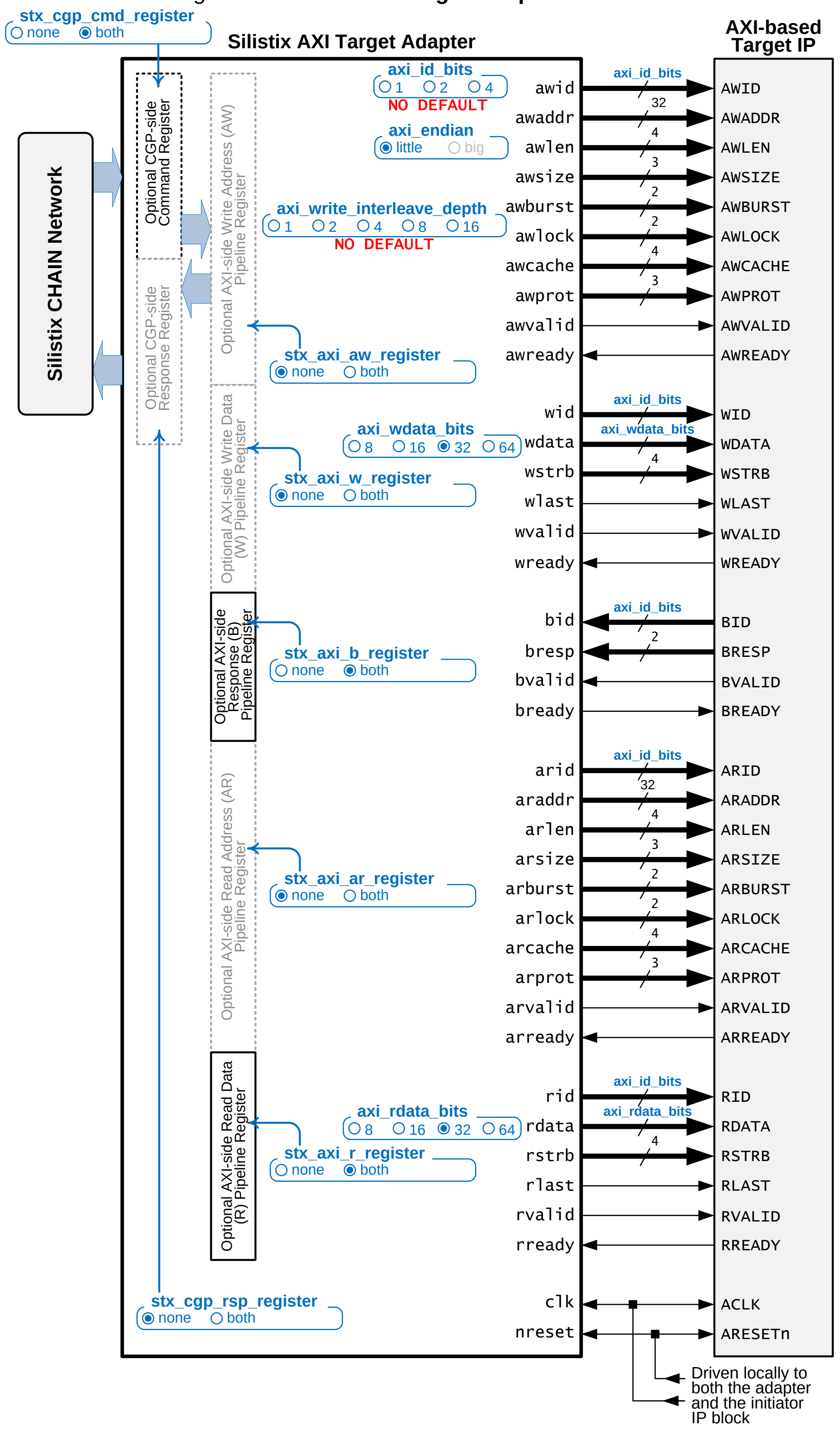

AHB-based Initiator IP
Silistix AXI Initiator Adapter
stx_cgp_cmd_register [✓ none] both
AWID
AWADDR
AWLEN
AWSIZE
AWBURST
AWLOCK
AWCACHE
AWPROT
AWVALID
AWREADY
axi_id_bits
32
4
3
2
2
4
3
awid
awaddr
awlen
awsize
awburst
awlock
awcache
awprot
awvalid
awready
axi_id_bits 1 2 4
NO DEFAULT
axi_endian [✓ little] big
stx_axi_aw_register none [✓ both]
Optional AXI-side Write Address (AW) Pipeline Register
Optional CGP-side Command Register
Optional CGP-side Response Register
Silistix CHAIN Network
WID
WDATA
WSTRB
WLAST
WVALID
WREADY
axi_id_bits
axi_wdata_bits
4
wid
wdata
wstrb
wlast
wvalid
wready
axi_wdata_bits 8 16 [✓ 32] 64
stx_axi_w_register none [✓ both]
Optional AXI-side Write Data (W) Pipeline Register
BID
BRESP
BVALID
BREADY
axi_id_bits
2
bid
bresp
bvalid
bready
stx_axi_b_register [✓ none] both
Optional AXI-side Response (B) Pipeline Register
stx_axi_max_outstanding [✓ 1] 2 4 8 16
stx_axi_max_allowable_ids [✓ 1] 2 4
stx_axi_max_allowable_id_instances [✓ 1] 2 4
ARID
ARADDR
ARLEN
ARSIZE
ARBURST
ARLOCK
ARCACHE
ARPROT
ARVALID
ARREADY
axi_id_bits
32
4
3
2
2
4
3
arid
araddr
arlen
arsize
arburst
arlock
arcache
arprot
arvalid
arready
stx_axi_ar_register none [✓ both]
Optional AXI-side Read Address (AR) Pipeline Register
RID
RDATA
RSTRB
RLAST
RVALID
RREADY
axi_id_bits
axi_rdata_bits
4
rid
rdata
rstrb
rlast
rvalid
rready
axi_rdata_bits 8 16 [✓ 32] 64
stx_axi_r_register [✓ none] both
Optional AXI-side Read Data (R) Pipeline Register
ACLK
ARESETn
clk
nreset
stx_cgp_rsp_register none [✓ both]
Driven locally to both the adapter and the initiator IP block

Figure 8: **Silistix AXI Target Adapter and Attributes**



The **ACLK** clock signal to the target IP block also drives the clocked logic within the AXI adapter. Similarly, the target adapter is reset by the active-Low **ARESETn** signal.

Optional Pipeline Register Control

As shown in Figure 7 and Figure 8, pipeline registers are optionally available for both the CGP-side and the AXI-side adapter interfaces. If inserted, then the register uses the same clock source as the AXI IP block.

AXI-side Interface

The AXI adapter provides optional pipeline registers for each of the AXI signal groups. Each register is individually controlled by an attribute as shown in Table 7. Setting the attribute to **both** inserts the pipeline registers into the associated signal group. Inserting these pipeline registers adds levels of latency.

To remove the pipeline registers, set the associated attribute to **none**.

Table 7: **AXI-side Pipeline Attributes**

Signal Group/Channel	Pipeline Register Control	Initiator Default Value	Target Default Value
Write Address	stx_axi_aw_register	both	none
Write Data	stx_axi_w_register	both	none
Write Response	stx_axi_b_register	none	both
Read Address	stx_axi_ar_register	both	none
Read Data	stx_axi_r_register	none	both

CGP-side Interface

The CGP-side interface has both a command path and a response path. The command path includes address, command, or write operations from the initiator to the selected target. The response path includes any read or status responses from the selected target back to the initiator.

The CGP interface provides optional pipeline registers for both the command and response paths. Each register is individually controlled by an attribute as shown in Table 8. Setting the attribute to **both** inserts the pipeline registers to the associated signal group. Inserting these pipeline registers adds levels of latency.

To remove the pipeline registers, set the associated attribute to **none**.

Table 8: **CCP-side Pipeline Attributes**

CGP Path	Pipeline Register Control	Initiator Default Value	Target Default Value
Command	stx_cgp_cmd_register	none	both
Response	stx_cgp_rsp_register	both	none

Initiator Outstanding Transactions Control

By default, the initiator adapter supports a single outstanding transaction. However, using the following advanced attributes, the initiator supports multiple unacknowledged outstanding transactions. Increasing the number of outstanding transactions generally improves performance but also increases the amount of silicon area consumed.

Allowable IDs

The AXI adapter `axi_id_bits` attribute defines how many unique AXI ID bits are supplied on the adapter interface. The `stx_axi_allowable_ids` advanced attribute specifies how many different AXI IDs are allowed to have simultaneous outstanding transactions.

Setting this attribute lower saves area, but also impairs potential bus performance.

```
stx_axi_allowable_ids = { 1 | 2 | 4 }
```

Allowable ID Instances

The `stx_axi_allowable_id_instances` advanced attribute specifies how many unique instances of a given AXI ID value are allowed to have simultaneous outstanding transactions.

Setting this attribute lower saves area, but also impairs potential bus performance.

```
stx_axi_allowable_id_instances = { 1 | 2 | 4 }
```

Maximum Number of Outstanding Transactions

The `stx_axi_max_outstanding` advanced attribute specifies the maximum number of outstanding commands that the initiator is allowed to generate, regardless of the AXI ID.

Setting this attribute lower saves area, but also impairs potential bus performance.

```
stx_axi_max_outstanding = { 1 | 2 | 4 | 8 | 16 }
```

Target Write Interleave Control

Write data interleaving allows a target adapter to accept interleaved write data transfers with different AWID values. The mandatory `axi_write_interleave_depth` attribute defines the write data interleaving depth that the adapter supports.

To avoid a deadlock condition, the `axi_write_interleave_depth` attribute must be no larger than the maximum amount of interleaved write data that the target IP can continuously accept.

```
axi_write_interleave_depth = { 1 | 2 | 4 | 8 | 16}
```

Table 9: **AXI Adapter Attributes Summary**

Attribute Name	Default	Allowable Values	Description
General Attributes			
axi_endian	`little`	`little`	Endian
axi_raddr_bits	32	32	Read address bus bit width
axi_waddr_bits	32	32	Write address bus bit width

Attribute Name	Default	Allowable Values	Description
axi_rdata_bits	32	**8, 16, 32, 64**	Read data bus bit width
axi_wdata_bits	32	**8, 16, 32, 64**	Write data bus bit width
axi_id_bits	No Default Must specify	**1, 2, 4**	Width of ID signals
Optional Pipeline Register Control			
stx_axi_aw_register	See Table 7	**none, both**	Optional AXI-side pipeline register, Write Address channel
stx_axi_w_register		**none, both**	Optional AXI-side pipeline register, Write Data channel
stx_axi_b_register		**none, both**	Optional AXI-side pipeline register, Write Response channel
stx_axi_ar_register		**none, both**	Optional AXI-side pipeline register, Read Address channel
stx_axi_r_register		**none, both**	Optional AXI-side pipeline register, Read Data channel
stx_cgp_cmd_register	See Table 8	**none, both**	Optional CGP-side command pipeline register
stx_cgp_rsp_register		**none, both**	Optional CGP-side response pipeline register
Initiator Outstanding Transactions Control			
stx_axi_allowable_ids	**1**	**1, 2, 4**	**Initiator only:** Number of bursts, with different AXI IDs, allowed to be outstanding
stx_axi_allowable_id_instances	**1**	**1, 2, 4**	**Initiator only:** Number of bursts, with same AXI ID, allowed to be outstanding
stx_axi_max_outstanding	**1**	**1, 2, 4, 8, 16**	**Initiator only:** Maximum number of outstanding bursts allowed, regardless of the AXI ID
Target Write Interleave Control			
axi_write_interleave_depth	No Default Must specify	**1, 2, 4, 8, 16**	**Target only:** Write interleave depth

More Information

The Silistix AXI adapters were created according the AXI specifications available under license from ARM.

- **AMBA 3 Specification**
  www.arm.com/products/solutions/axi_spec.html
- **AMBA 2 Specification**
  www.arm.com/products/solutions/AMBA_Spec.html

Open Core Protocol (OCP) Adapter

The Silistix OCP adapter is designed according to the Open Core Protocol (OCP) Specification, Release 2.2. Generally the OCP initiator and target adapters support similar features, although there are differences between the two as outlined below.

IP cores designed to OCP Release 1.0 are not supported.

The Silistix OCP attribute names are identical to those used in the OCP specification, except that they have an “`ocp_`” prefix.

OCP Signal Groups

The OCP interface specification divides the interface signals into a few simple categories, listed below. This chapter is similarly divided.

The OCP signal groups include the following.

- **Basic Signals**, which are generally required for any OCP interface.
- **Simple Extensions** that add more control signals and provide more information about each transfer.
- **Burst Extensions**, which allow grouping of multiple transfers that have a defined address relationship.
- **Tag Extensions** that tag data transfers for out-of-order responses and write data commit.
- **Thread Extensions** that add support for multi-threading of the OCP interface.
- **Sideband Signals** that contain signals that are not part of the dataflow phases.
- **Test Signals** that add support for scan, clock control, and IEEE 1149.1 (JTAG).

OCP Attributes

The Silistix OCP adapters are designed with the following OCP attribute settings.

Protocol Attributes

- The supported command enable attributes are described in the subsequent sections describing each OCP signal group
- `ocp_burst_aligned = 0`: There are no restrictions on INCR bursts. This is the OCP default.
- See “Burst Extensions” on page 38 for the supported burst sequence enable attributes.
- `ocp_endian = little`: This is the OCP default.
- `ocp_force_aligned = 0`: There are no restrictions on byte-enable signals. This is the OCP default.

- `ocp_mthreadbusy_exact = 0`, `ocp_sdatathreadbusy_exact = 0`, and `ocp_sthreadbusy_exact = 0`: The Silistix OCP adapters currently do not support multiple threads. This is the OCP default.
- `ocp_tag_interleave_size = 1`: There are no restrictions on response interleaving. This is the OCP default.

Phase Attributes

- `ocp_datahandshake = 1`: The datahandshake phase is always present. This is NOT the OCP default.
- `ocp_reqdata_together = 0`: The first datahandshake phase is independent of the request phase. The initiator adapter may receive the first datahandshake phase in the same cycle as the request phase, or in a later cycle. The target adapter may issue the first datahandshake phase in the same cycle as the request phase, or in a later cycle.
- `ocp_writeresp_enable`: Both 0 and 1 are supported. The adapters optionally have responses enabled on write-type commands.

Signal Attributes

The signal attribute settings are described in the subsequent sections describing each OCP signal group.

Basic Signals

Generally OCP-based designs use most of the signals from the OCP basic signal group. The Silistix OCP adapters, shown in Figure 9 and Figure 10, support the basic signal group with the following characteristics and restrictions.

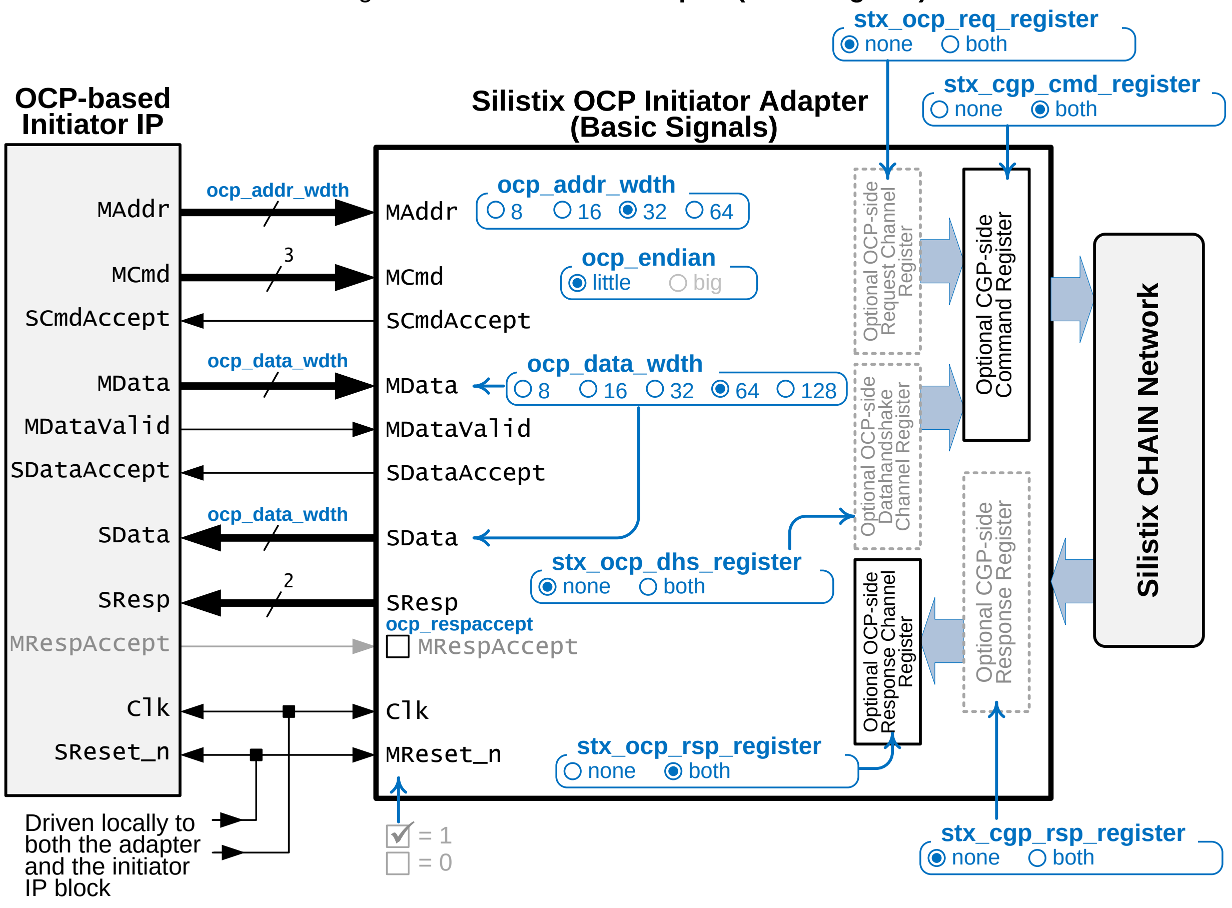
- The `Clk` signal is mandatory according to the OCP specification.
- The `EnableClk` signal is not supported. Consequently the attribute `ocp_enableclk = 0`.
- The `MAddr` signal is is supported and is mandatory (`ocp_addr = 1`) for both initiator and target adapters. The behavior of the initiator adapter is undefined if it is issued a request with an unsupported address value on the `MAddr` signal. The initiator IP block must ensure that only supported addresses are issued to the adapter. Furthermore, the initiator IP block must align addresses presented on the `MAddr` signal to the data-word size, forcing the necessary numbers of lesser-significant address bits to zero.
- The `MCmd` signal is mandatory according to the OCP specification. Presently the OCP adapters support the IDLE, WR and RD commands on the `MCmd` signal. The initiator IP block must ensure that only the supported commands are issued to the adapter. The behavior of the initiator adapter is undefined if it receives an unsupported command. Furthermore, the OCP adapters must support both WR and RD commands. Therefore the only legal setting for the command enable attributes are as follows

`ocp_broadcast_enable`	0
`ocp_rdlwrc_enable`	0
`ocp_read_enable`	1
`ocp_readex_enable`	0
`ocp_write_enable`	1
`ocp_writenonpost_enable`	0

- The `MData` signal is supported and is mandatory (`ocp_mdata = 1`) for both initiator and target adapters.
- The `MDataValid` signal is supported and is mandatory (`ocp_datahandshake = 1`) for both initiator and target adapters.
- The `MRespAccept` signal is supported. It is optional (`ocp_respaccept = 0` or `1`) for the initiator adapter and mandatory (`ocp_respaccept = 1`) for the target adapter.
- The `SCmdAccept` signal is supported. It is mandatory (`ocp_cmdaccept = 1`) for the initiator adapter and optional (`ocp_scmdaccept = 0` or `1`) for the target adapter.
- The `SData` signal is supported and is mandatory (`ocp_sdata = 1`) for both initiator and target adapters.
- The `SDataAccept` signal is supported. It is mandatory (`ocp_dataaccept = 1`) for the initiator adapter and optional (`ocp_dataaccept = 0` or `1`) for the target adapter.
- The `SResp` signal is supported and is mandatory (`ocp_resp = 1`) for both initiator and target adapters. An OCP error is defined as when `SResp` = ERR in a response phase. The initiator adapter will NOT respond with an OCP error in the event of an unsupported request phase. It will only respond with `SResp` = ERR if the error was generated by the target IP block on the other side of the network. Supported encodings of `SResp` are NULL, DVA and ERR. The target IP block must ensure that only supported responses are issued to the target adapter.

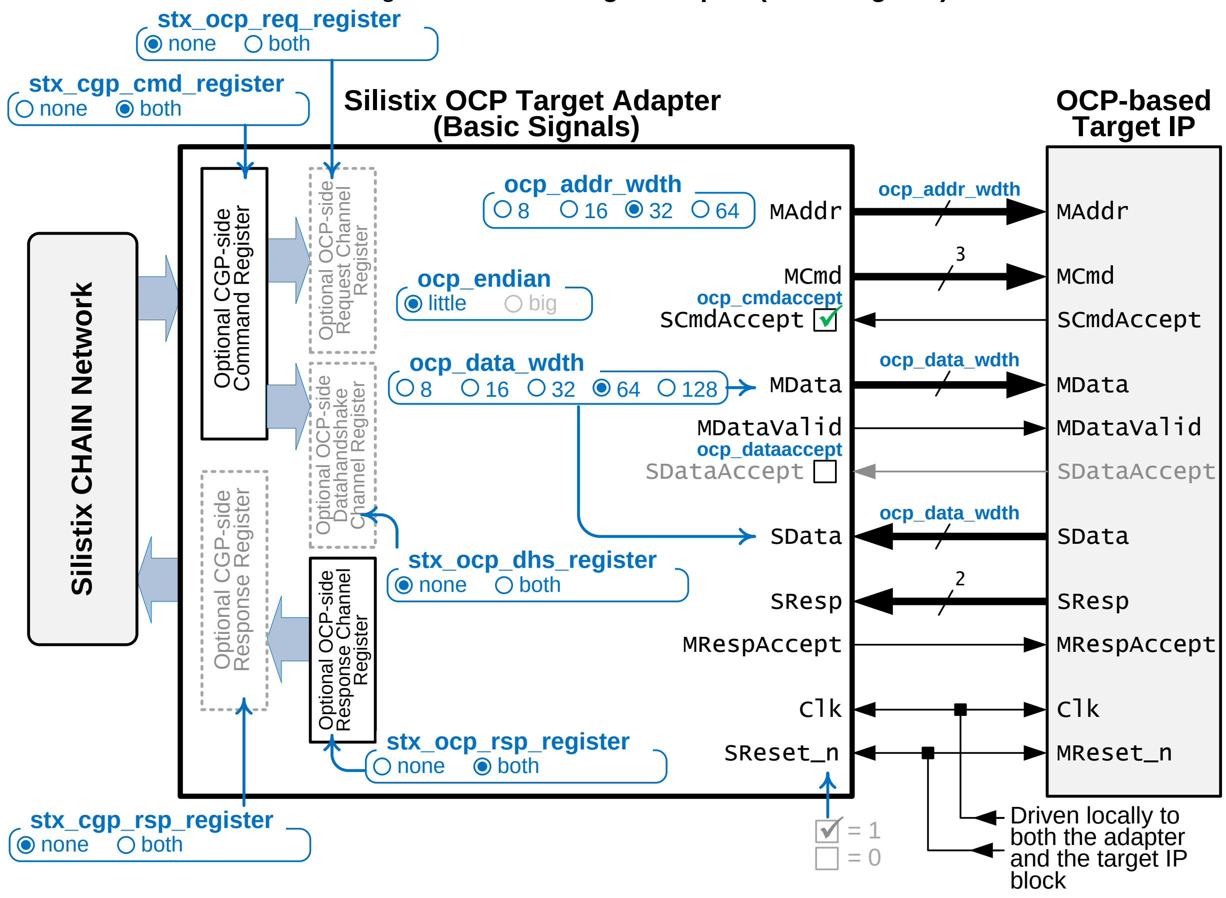
The basic signals on an OCP initiator adapter appear in Figure 9. The `MRespAccept` signal is the only optional signal for the initiator adapter. The attribute name to enable or disable a particular interface signal appears directly above the associated check box in Figure 9.

Figure 9: **OCP Initiator Adapter (Basic Signals)**



The basic signals on an OCP target adapter appear in Figure 10. The **`SCmdAccept`** and **`SDataAccept`** signals are optional for the target adapter; all other signals are mandatory. The attribute name to enable or disable a particular interface signal appears directly above the associated check box in Figure 10.

Figure 10: **OCP Target Adapter (Basic Signals)**



General Attributes

The following attributes are those that apply to all OCP adapter applications. The name of each attribute is listed along with the allowable values. The default setting for the attribute appears in **<u>`Bold Underline`</u>**.

Endian

The current OCP adapters only support little endian data ordering.

`ocp_endian` = { **<u>`little`</u>** }

Basic Interface Signals

Some of the adapter basic interface signals are optional, controlled by individual enable or disable attributes as shown in Table 10.

Table 10: **Basic Signal Attributes**

Interface Signal	Silistix Attribute	OCP Parameter	Initiator Default Value	Target Default Value
SCmdAccept	`ocp_cmdaccept`	`cmdaccept`	1 (mandatory)	1 (optional)
SDataAccept	`ocp_dataaccept`	`dataaccept`	1 (mandatory)	0 (optional)
MRespAccept	`ocp_respaccept`	`respaccept`	0 (optional)	1 (mandatory)

Address Bus Width

The **`ocp_addr_wdth`** attribute defines the width of the OCP address bus, measured in bits.

`ocp_addr_wdth = { 8 | 16 | 32 | 64 }`

Data Bus Width

The OCP interface has separate read and write data busses but both have the same width. The **`ocp_data_wdth`** attribute defines the data bus bit width.

`ocp_data_wdth = { 8 | 16 | 32 | 64 | 128 }`

Optional Pipeline Register Control

As shown in Figure 10, pipeline registers are optionally available for both the CGP-side and the OCP-side adapter interfaces. If inserted the register uses the same clock source as the OCP IP block.

OCP-side Interface

The OCP-side interface has separate Request , Response , and Datahandshake channels, each with individually-controlled, optional pipeline registers. The signals associated with each channel are listed in Table 11. Depending on other attribute settings, not all of the signals listed may be present on the interface.

Request Channel

By default there is no additional pipelining registers on the OCP request channel. Set **`stx_ocp_req_register = both`** to add a pipeline register to the inputs and outputs of the OCP-side interface. Inserting these pipeline registers adds levels of latency.

`stx_ocp_req_register = { none | both }`

Datahandshake Channel

By default there are no pipelining registers present on the OCP datahandshake channel. Set **`stx_ocp_dhs_register = both`** to add a pipeline register to the inputs and outputs of the OCP-side interface. Inserting these pipeline registers adds levels of latency.

`stx_ocp_dhs_register = { none | both }`

Response Channel

By default there are pipelining registers on both the inputs to and outputs from the OCP response channel. These pipeline registers add additional levels of latency. Set **`stx_ocp_rsp_register = none`** to remove the pipeline registers.

`stx_ocp_rsp_register = { none | both }`

Table 11: **OCP Phase Groups and Associated Signals on the Silistix Adapters**

Request Group	Datahandshake Group	Response Group
MAddr	MData	SData
MAddrSpace	MDataByteEn	SDataInfo
MBurstLength	MDataInfo	SResp
MBurstSeq	MDataLast	SRespInfo
MBurstSingleReq	MDataTagID	SRespLast
MByteEn	MDataValid	STagID
MConnID		
MCmd		
MReqInfo		
MTagID		

CGP-side Interface

The CGP-side interface has both a command path and a response path. The command path includes address, command, and write operations from the initiator to the selected target. The response path includes any read or status responses from the selected target back to the initiator.

Command Path

By default there are additional pipelining registers on the inputs and outputs of the CGP command interface signals. These pipeline registers add additional levels of latency. Set `stx_cgp_cmd_register = none` to remove the pipeline registers.

```
stx_cgp_cmd_register = { none | both }
```

Response Path

By default there are no additional pipelining registers on the CGP response interface signals. Set `stx_cgp_rsp_register = both` to add a pipeline register to the inputs and outputs of the CGP-side response interface. Inserting these pipeline registers adds levels of latency.

```
stx_cgp_rsp_register = { none | both }
```

Simple Extensions

The OCP initiator and target adapters, shown in Figure 11 and Figure 12, support the simple extensions defined in the OCP specification, also listed in Table 12. None of the simple extension signals are enabled by default and must be individually enabled using associated attributes.

- The `MAddrSpace` signal is supported and is optional (`ocp_addrspace = 0` or `1`) for both initiator and target adapters.
- The `MByteEn` signal is supported and is optional (`ocp_byteen = 0` or `1`) for both initiator and target adapters.
- The `MDataByteEn` signal is supported and is optional (`ocp_mdatabyteen = 0` or `1`) for both initiator and target adapters.
- The `MDataInfo` signal is supported and is optional (`ocp_mdatainfo = 0` or `1`) for both initiator and target adapters.
- The `MReqInfo` signal is supported and is optional (`ocp_reqinfo = 0` or `1`) for both initiator and target adapters.

- The **SDataInfo** signal is supported and is optional (**ocp_sdatainfo = 0** or **1**) for both initiator and target adapters.
- The **SRespInfo** signal is supported and is optional (**ocp_respinfo = 0** or **1**) for both initiator and target adapters.

The width of the **MByteEn** and **MDataByteEn** byte enable signals, if they are present on the interface, is defined by the **ocp_data_wdth** attribute divided by 8. In other words there is one individual byte enable signal for each byte in the data word.

The width of the other extension signals is controlled by associated, individual attributes as listed in Table 12.

Figure 11: **OCP Initiator Adapter Simple Extensions**

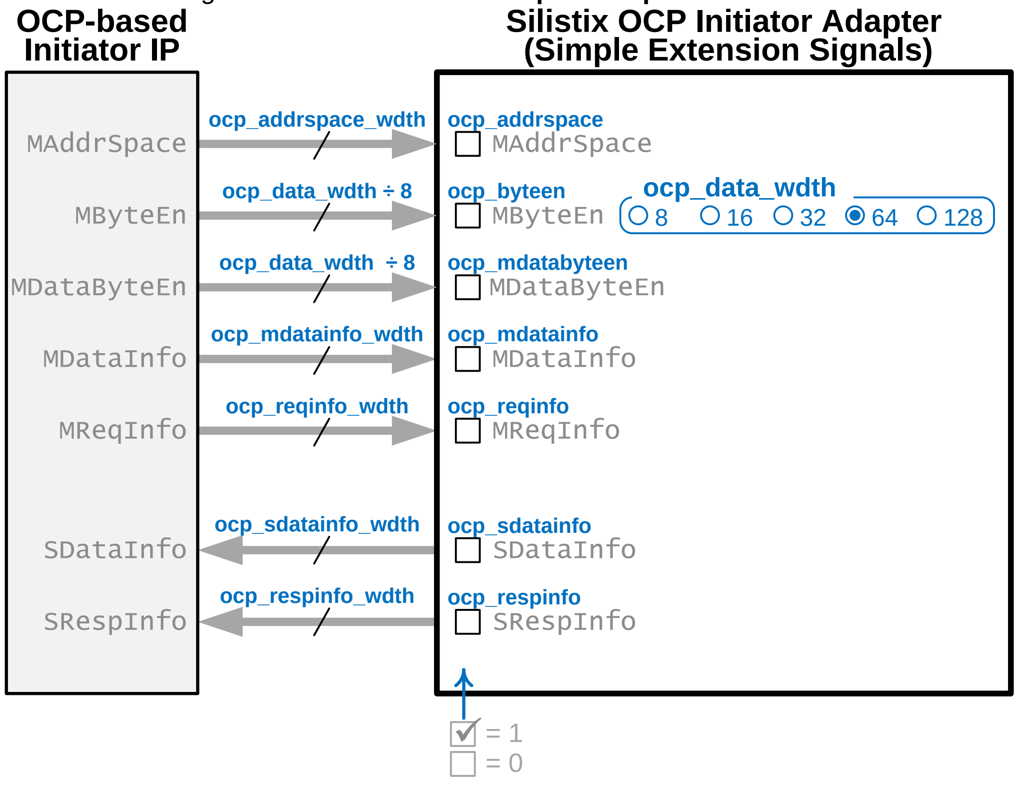


Figure 12: **OCP Target Adapter Simple Extensions**

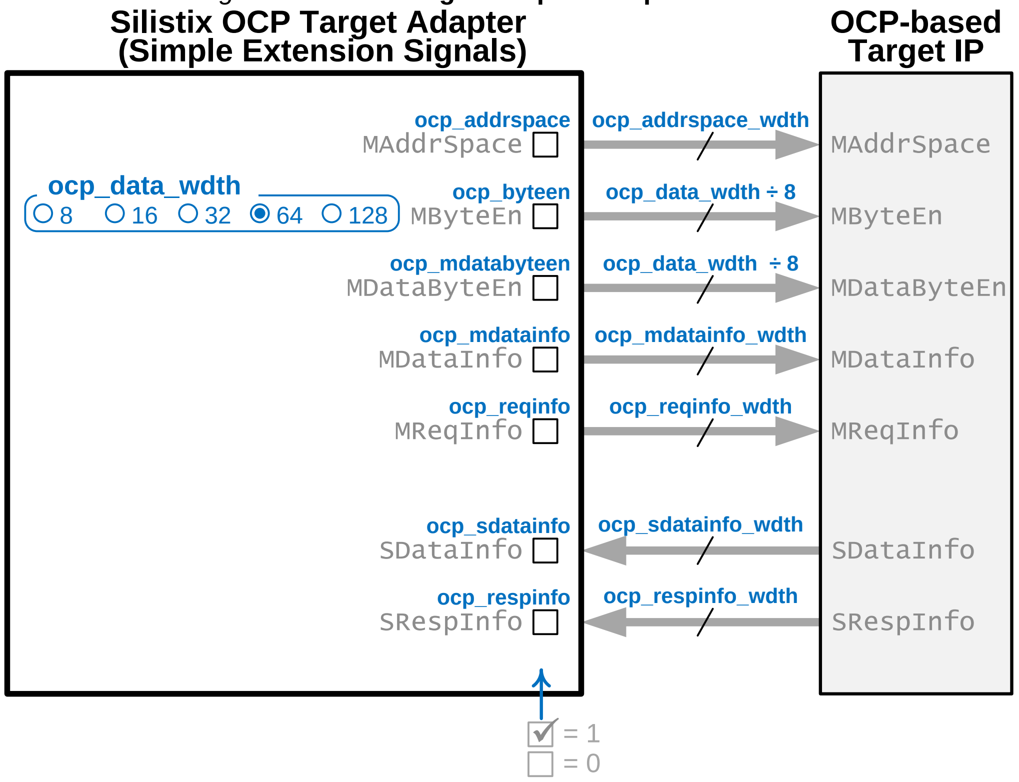


Table 12: **Simple Extensions Signal Attributes**

Interface Signal	Silistix Attribute	OCP Parameter	Related Control(s)	Initiator Default Value	Target Default Value
MAddrSpace	ocp_addrspace	addrspace	ocp_addrspace_wdth	0	0
MByteEn	ocp_byteen	byteen	ocp_data_wdth	0	0
MDataByteEn	ocp_mdatabyteen	mdatabyteen	ocp_data_wdth	0	0
MDataInfo	ocp_mdatainfo	mdatainfo	ocp_mdatainfo_wdth, ocp_mdatainfobyte_wdth	0	0
MReqInfo	ocp_reqinfo	reqinfo	ocp_reqinfo_wdth,	0	0
SDataInfo	ocp_sdatainfo	sdatainfo	ocp_sdatainfo_wdth, ocp_sdatainfobyte_wdth	0	0
SRespInfo	ocp_respinfo	respinfo	ocp_respinfo_wdth	0	0

Address Space Width

To add the **`MAddrSpace`** signal in the adapter interface, set the attribute **`ocp_addrspace = 1`** and the **`ocp_addrspace_wdth`** attribute to an integer > 0.

MDataInfo Width

To add the **`MDataInfo`** signal in the adapter interface, set the attribute **`ocp_mdatainfo = 1`** and the **`ocp_mdatainfo_wdth`** attribute to an integer > 0. The restriction on the value of **`ocp_mdatainfo_wdth`** attribute in relation to **`ocp_mdatainfobyte_wdth`** attribute must be obeyed (see the OCP specification for additional information).

MReqInfo Width

To add the `MReqInfo` signal in the adapter interface, set the attribute `ocp_reqinfo = 1` and the `ocp_reqinfo_wdth` attribute to an integer > 0.

SDataInfo Width

To add the `SDataInfo` signal in the adapter interface, set the attribute `ocp_sdatainfo = 1` and the `ocp_sdatainfo_wdth` attribute to an integer > 0. The restriction on the value of `ocp_sdatainfo_wdth` attribute in relation to `ocp_sdatainfobyte_wdth` attribute must be obeyed (see the OCP specification for additional information).

SRespInfo Width

To add the `SRespInfo` signal in the adapter interface, set the attribute `ocp_respinfo = 1` and the `ocp_respinfo_wdth` attribute to an integer > 0.

Burst Extensions

The Burst Extension signals are supported on both the OCP initiator adapter (Figure 13) and the OCP target adapter (Figure 14), with the following restrictions and capabilities.

- The `MAtomicLength` signal is NOT supported (`ocp_atomiclength = 0`).
- The `MBlockHeight` signal is NOT supported (`ocp_blockheight = 0`).
- The `MBlockStride` signal is NOT supported (`ocp_blockstride = 0`).
- The `MBurstLength` signal is supported and is optional (`ocp_burstlength = 0` or `1`) for both initiator and target adapters.
- The `MBurstPrecise` signal is NOT supported (`ocp_burstprecise = 0`). All burst transactions are precise by default.
- The `MBurstSeq` signal is supported and is optional (`ocp_burstseq = 0` or `1`) for both initiator and target adapters. Only the INCR, STRM, WRAP and XOR burst sequences are supported on `MBurstSeq`. An initiator IP block must ensure that only a supported burst sequence is issued to the initiator adapter. The behaviour of the initiator adapter is undefined if it receives an unsupported burst sequence. Legal settings for the burst sequence enable attributes are as follows

`ocp_burstseq_blck_enable`	`0`
`ocp_burstseq_dflt1_enable`	`0`
`ocp_burstseq_dflt2_enable`	`0`
`ocp_burstseq_incr_enable`	`0` or `1`
`ocp_burstseq_strm_enable`	`0` or `1`
`ocp_burstseq_unkn_enable`	`0`
`ocp_burstseq_wrap_enable`	`0` or `1`
`ocp_burstseq_xor_enable`	`0` or `1`

- The `MBurstSingleReq` signal is supported and is optional (`ocp_burstsinglereq = 0` or `1`) for both initiator and target adapters. However the initiator IP block must always drive `MBurstSingleReq` high during the request phase if it is present on the initiator adapter. The target adapter always drives `MBurstsingleReq` high during the request phase if it is present. If the adapter does not have `MBurstSingleReq` then the adapter applies a tie-off value of 1 for this signal. In other words the adapters only support single-request/multiple-data burst transactions.

- The `MDataLast` signal is supported. It is mandatory (`ocp_datalast = 1`) for the initiator adapter if burst transactions are supported and optional (`ocp_datalast = 0` or `1`) for the target adapter.
- The `MDataRowLast` signal is NOT supported (`ocp_datarowlast = 0`).
- The `MReqLast` signal is NOT supported (`ocp_reqlast = 0`).
- The `MReqRowLast` signal is NOT supported (`ocp_reqrowlast = 0`).
- The `SRespLast` signal is supported. It is optional (`ocp_resplast = 0` or `1`) for the initiator adapter and mandatory (`ocp_resplast = 1`) for the target adapter if burst transactions are supported.
- The `SRespRowLast` signal is NOT supported (`ocp_resprowlast = 0`).

Figure 13: **OCP Initiator Adapter Burst Signals**

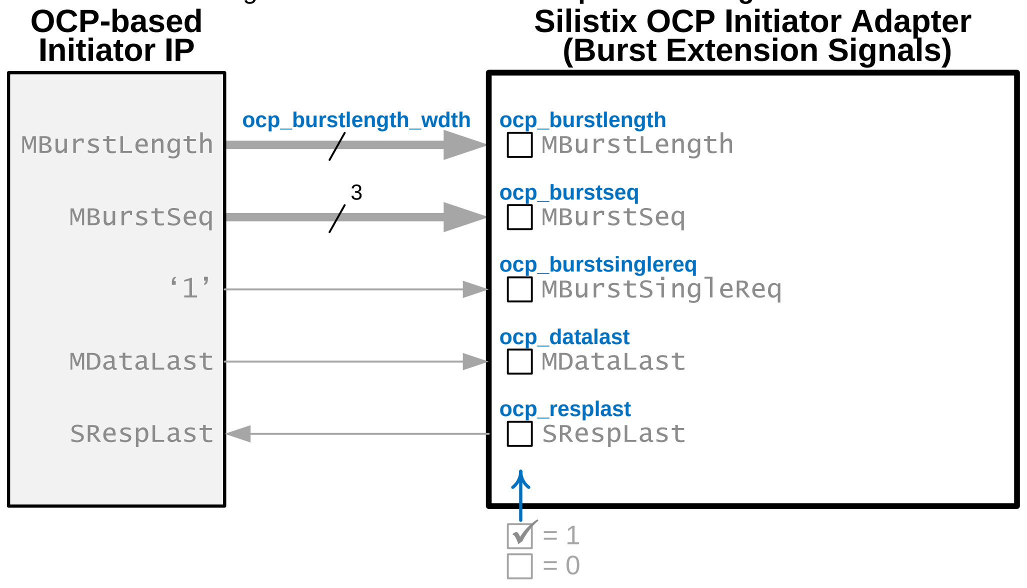


Figure 14: **OCP Target Adapter Burst Signals**

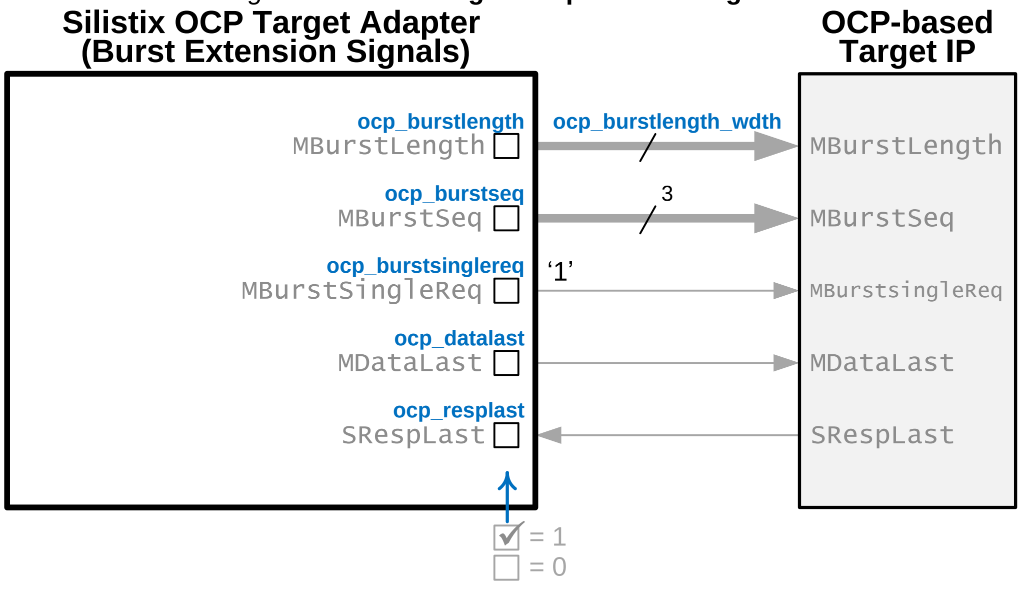


Table 13 lists the various burst extension signals on the initiator and target adapters and the associated attributes. Set the signal attribute to 1 to include the signal on the interface. Leave it at the default value 0 to remove the signal from the interface.

If the attribute `ocp_burstlength=1`, then also define a value for the associated width control attribute, `ocp_burstlength_wdth`, which sets the bit width of the `MBurstLength` signal. The `ocp_burstlength_wdth` attribute must be an integer, 1 or greater.

Table 13: **Burst Extensions Signal Attributes**

Interface Signal	Silistix Attribute	OCP Parameter	Related Control(s)	Initiator Default Value	Target Default Value
MBurstLength	`ocp_burstlength`	`burstlength`	`ocp_burstlength_wdth`	0	0
MBurstSeq	`ocp_burstseq`	`burstseq`	N/A	0	0
MBurstSingleReq	`ocp_burst_singlereq`	`burst_singlereq`	`ocp_burstsinglereq_tieoff = 1`	0	0
MDataLast	`ocp_datalast`	`datalast`	N/A	0	0
SRespLast	`ocp_reslast`	`reslast`	N/A	0	0

Tag Extensions

Tag extensions are supported for both the initiator adapter, shown in Figure 15, and the target adapter, shown in Figure 16, with the following restrictions and capabilities.

- The `MDataTagID` signal is supported and is optional (`ocp_tags = 1` or greater) for both initiator and target adapters.
- The `MTagID` signal is supported and is optional (`ocp_tags = 1` or greater) for both initiator and target adapters.
- The `MTagInOrder` signal is NOT supported (`ocp_taginorder = 0`).
- The `STagID` signal is supported and is optional (`ocp_tags = 1` or greater) for both initiator and target adapters.

- The `STagInOrder` signal is NOT supported (`ocp_taginorder = 0`).

Figure 15: **OCP Initiator Adapter Tag Extension Signals**

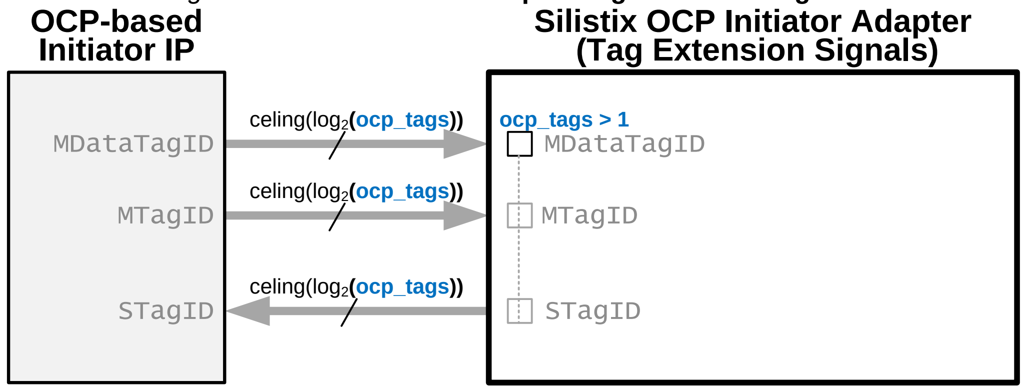
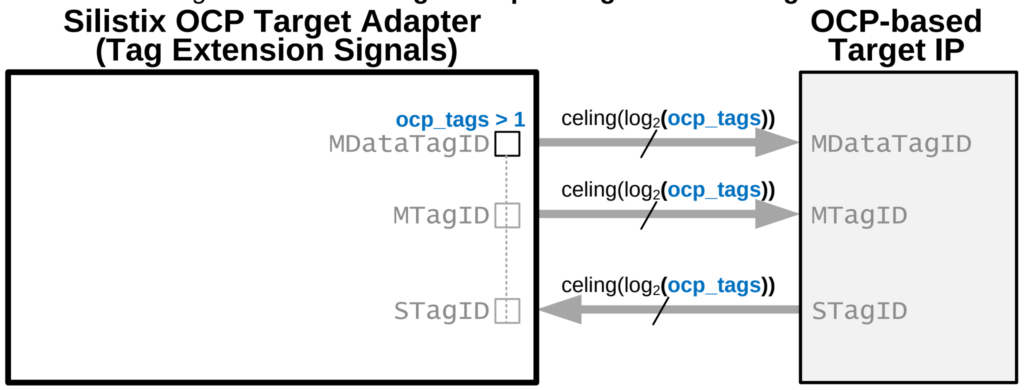


Figure 16: **OCP Target Adapter Tag Extension Signals**



All the tag signals are controlled by a single attribute, `ocp_tags`, as shown in Table 14. If the `ocp_tags` attribute is greater than 1, then all three of the supported tag signals are added to the interface. The width of each signal becomes the number of wires required to represent the number of possible tag states defined by the `ocp_tags` attribute using binary encoding.

Table 14: **Tag Extensions Signal Attributes**

Interface Signal	Silistix Attribute	OCP Parameter	Width Control	Initiator Default Value	Target Default Value
MDataTagID	ocp_tags	tags	ceiling[log2(ocp_tags)]	1	1
MTagID					
STagID					

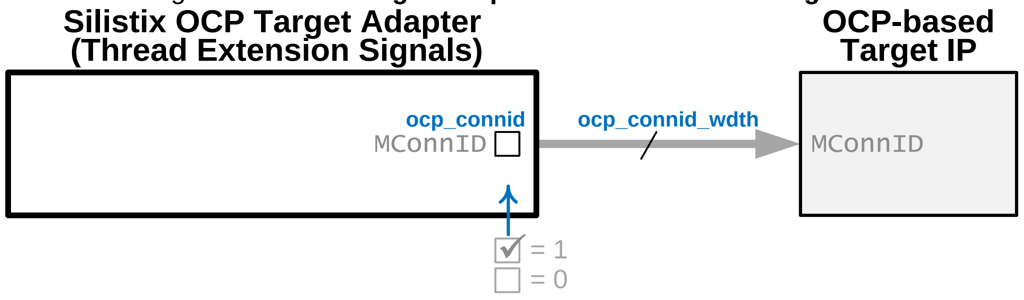
Thread Extensions

Thread extensions have limited support in the OCP target adapter, shown in Figure 17. There is no thread support for the OCP initiator adapter.

- The `MConnID` signal is supported by the target adapter and is optional (`ocp_connid = 0` or `1`). The initiator adapter does NOT support `MConnID` (`ocp_mconnid = 0`).
- The `MDataThreadID` signal is NOT supported (`ocp_threads = 1`).

- The `MThreadBusy` signal is NOT supported (`ocp_mthreadbusy = 0`).
- The `MThreadID` signal is NOT supported (`ocp_threads = 1`).
- The `SDataThreadBusy` signal is NOT supported (`ocp_sdatathreadbusy = 0`).
- The `SThreadBusy` signal is NOT supported (`ocp_sthreadbusy = 0`).
- The `SThreadID` signal is NOT supported (`ocp_threads = 1`).

Figure 17: **OCP Target Adapter Thread Extension Signals**



To add the **`MConnID`** signal in the target adapter interface, set the attribute `ocp_connid = 1` and the `ocp_connid_wdth` attribute to an integer > 0. As shown in Table 15, there is no similar support in the initiator adapter.

Table 15: **Thread Extensions Signal Attributes**

Interface Signal	Silistix Attribute	OCP Parameter	Width Control	Initiator Default Value	Target Default Value
MConnID	ocp_connid	connid	ocp_connid_wdth	N/A	0

Sideband Signals

The OCP adapters only support the asynchronous, active-Low reset inputs as described below. The relevant reset signal appears in Figure 9 or Figure 10.

- The `Control` signal is NOT supported (`ocp_control = 0`).
- The `ControlBusy` signal is NOT supported (`ocp_controlbusy = 0`).
- The `ControlWr` signal is NOT supported (`ocp_controlwr = 0`).
- The `MError` signal is NOT supported (`ocp_merror = 0`).
- The `MFlag` signal is NOT supported (`ocp_mflag = 0`).
- The `MReset_n` signal is supported by the initiator adapter and is mandatory (`ocp_mreset = 1`). The target adapter does NOT support `MReset_n` (`ocp_mreset = 0`).
- The `SError` signal is NOT supported (`ocp_serror = 0`).
- The `SFlag` signal is NOT supported (`ocp_sflag = 0`).
- The `SInterrupt` signal is NOT supported (`ocp_interrupt = 0`).
- The `SReset_n` signal is supported by the target adapter and is mandatory (`ocp_sreset = 1`). The initiator adapter does NOT support `SReset_n` (`ocp_sreset = 0`).
- The `Status` signal is NOT supported (`ocp_status = 0`).

- The `StatusBusy` signal is NOT supported (`ocp_statusbusy = 0`).
- The `StatusRd` signal is NOT supported (`ocp_statusrd = 0`).

Table 16 lists the associated attributes and their default values for the initiator and target adapters.

Table 16: **Sideband Signal Attributes**

Interface Signal	Silistix Attribute	OCP Parameter	Related Control	Initiator Default Value	Target Default Value
MReset_n	`ocp_mreset`	`mreset`	N/A	1	0
SReset_n	`ocp_sreset`	`sreset`	N/A	0	1

Test Signals

The OCP adapters do not support any test signals.

- The `ClkByp` signal is NOT supported (`ocp_clkctrl_enable = 0`).
- The `ScanCtrl` signal is NOT supported (`ocp_scanport = 0`).
- The `Scanin` signal is NOT supported (`ocp_scanport = 0`).
- The `Scanout` signal is NOT supported (`ocp_scanport = 0`).
- The `TCK` signal is NOT supported (`ocp_jtag_enable = 0`).
- The `TDI` signal is NOT supported (`ocp_jtag_enable = 0`).
- The `TDO` signal is NOT supported (`ocp_jtag_enable = 0`).
- The `TestClk` signal is NOT supported (`ocp_clkctrl_enable = 0`).
- The `TMS` signal is NOT supported (`ocp_jtag_enable = 0`).
- The `TRST_N` signal is NOT supported (`ocp_jtagtrst_enable = 0`).

OCP Core RTL Configuration Files

To be compliant with the OCP-IP specification, a core and its OCP interfaces must be described by an RTL configuration file. The syntax for this file is specified in "*Chapter 6: Core RTL Configuration File*," found in the Open Core Protocol Specification, Release 2.2. Further details on each statement can be found in the specification.

Several example OCP core RTL configuration files are provided below.

- Default Initiator Adapter
- Default Target Adapter
- Initiator Adapter for MIP32-24K Processor

Default Initiator Adapter

Figure 18 shows the OCP core RTL configuration file for the default initiator adapter. This contains the minimum level of the OCP protocol that must be supported.

Figure 18: **Core RTL Configuration File: Default Initiator Adapter**

```
version 4.5

module slocp_adapt_ocp_cgp_pmf {

     core_id <vendor_code> <core_code> <revision_code> [<description>]

     interface ocp bundle ocp2 revision 2 {

            interface_type slave

            port clk                    net Clk
            port nreset                 net MReset_n
            port ocp_MAddr              net MAddr
            port ocp_MCmd               net MCmd
            port ocp_MData              net MData
            port ocp_MDataValid         net MDataValid
            port ocp_SCmdAccept         net SCmdAccept
            port ocp_SData              net SData
            port ocp_SDataAccept        net SDataAccept
            port ocp_SResp              net SResp

            prefix “”

            param datahandshake         1

            param addr                  1      {width 32}
            param burstsinglereq        0      {tie_off 1}
            param cmdaccept             1
            param dataaccept            1
            param data_wdth             64
            param mdata                 1
            param resp                  1
            param sdata                 1

            param mreset                1
            param sreset                0

     } # End of interface ocp

} # End of module slocp_adapt_ocp_cgp_pmf
```

Default Target Adapter

Figure 19 shows the OCP core RTL configuration file for the default target adapter. This contains the minimum level of the OCP protocol that must be supported.

Figure 19: **Core RTL Configuration File: Default Target Adapter**

```
version 4.5

module slocp_adapt_cgp_pmf_ocp {

     core_id <vendor_code> <core_code> <revision_code> [<description>]

     interface ocp bundle ocp2 revision 2 {

            interface_type master

            port clk                  net Clk
            port nreset               net SReset_n
            port ocp_MAddr            net MAddr
            port ocp_MCmd             net MCmd
            port ocp_MData            net MData
            port ocp_MDataValid       net MDataValid
            port ocp_MRespAccept      net MRespAccept
            port ocp_SCmdAccept       net SCmdAccept
            port ocp_SData            net SData
            port ocp_SResp            net SResp

            prefix “”

            param datahandshake       1

            param addr                1      {width 32}
            param burstsinglereq      0      {tie_off 1}
            param cmdaccept           1
            param data_wdth           64
            param mdata               1
            param resp                1
            param respaccept          1
            param sdata               1

            param mreset              0
            param sreset              1

     } # End of interface ocp

} # End of module slocp_adapt_cgp_pmf_ocp
```

Initiator Adapter for MIP32-24K Processor

Figure 20 shows the OCP core RTL configuration file for an initiator adapter that will connect to the OCP master system interface of the MIPS32-24K processor family.

Figure 20: **Core RTL Configuration File: Initiator Adapter for MIPS32-24K Processor**

```
version 4.5
module slocp_adapt_ocp_cgp_pmf {
     core_id <vendor_code> <core_code> <revision_code> [<description>]
     interface ocp bundle ocp2 revision 2 {
            interface_type slave

            port clk                  net Clk
            port nreset               net MReset_n
            port ocp_MAddr            net MAddr
            port ocp_MAddrSpace       net MAddrspace
            port ocp_MBurstLength     net MBurstLength
            port ocp_MBurstSeq        net MBurstSeq
            port ocp_MByteEn          net MByteEn
            port ocp_MCmd             net MCmd
            port ocp_MData            net MData
            port ocp_MDataByteEn      net MDataByteEn
            port ocp_MDataLast        net MDataLast
            port ocp_MDataTagID       net MDataTagID
            port ocp_MDataValid       net MDataValid
            port ocp_MReqInfo         net MReqInfo
            port ocp_MTagID           net MTagID
            port ocp_SCmdAccept       net SCmdAccept
            port ocp_SData            net SData
            port ocp_SDataAccept      net SDataAccept
            port ocp_SResp            net SResp
            port ocp_SRespInfo        net SRespInfo
            port ocp_SRespLast        net SRespLast
            port ocp_STagID           net STagID

            prefix “”

            param burstseq_incr_enable 0
            param burstseq_wrap_enable 1
            param burstseq_xor_enable  1

            param datahandshake        1

            param addr                 1      {width 32}
            param addrspace            1      {width 2}
            param burstlength          1      {width 3}
            param burstseq             1
            param burstsinglereq       0      {tie_off 1}
            param byteen               1
            param cmdaccept            1
            param dataaccept           1
            param datalast             1
            param data_wdth            64
            param mdata                1
            param mdatabyteen          1
            param reqinfo              1      {width 4}
            param resp                 1
            param respinfo             1      {width 1}
            param resplast             1
            param sdata                1
            param tags                 8

            param mreset               1
            param sreset               0

     } # End of interface ocp

} # End of module slocp_adapt_ocp_cgp_pmf
```

More Information

The Silistix OCP adapters were created according to the following specification, available under license from the Open Core Protocol International Partnership (OCP-IP).

- **OCP-IP Open Core Protocol Specification, Release 2.2**

More information about the Open Core Protocol (OCP) is available from the following web site.

- **Open Core Protocol International Partnership (OCP-IP)**
  www.ocpip.org

Silistix®

Glossary

Term	Description
AHB	AMBA High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	AMBA Peripheral Bus
AXI	AMBA Advanced eXtensible Interface
CGP	CHAIN Gateway Protocol
DMA	Direct Memory Access
IP	Intellectual Property
MIPS	MIPS Technologies
NoC	Network on Chip
OCP	Open Core Protocol
OCP-IP	Open Core Protocol International Partnership
PMF	Protocol Mapping Format
RTL	Register Transfer Level
SoC	System on Chip
Tcl	Tool Command Language

Revision History

Revision	Date	Description/Revisions
1.1.3	30-OCT-2008	Corrected settings for AXI_ID.
1.1.2	4-AUG-2008	Minor spelling corrections.
1.1.1	30-JUN-2008	Added Table of Contents. Clarified meaning of **`ocp_tags`** attribute.
1.1	22-MAY-2008	Corrected various typos and updated figures.
1.0	19-MAY-2008	Initial release.

Feedback

Feedback on this Silistix document and all Silistix products is highly encouraged. If you have a comment, correction, or suggestion to improve this document, please send us an E-mail. Please include complete details including page numbers, section titles, or figure or table numbers where appropriate. Thank you in advance for helping us to improve our products and services.

`feedback@silistix.com`