

Dynamic Function eXchange Decoupler v1.0

LogiCORE IP Product Guide

Vivado Design Suite

PG375 (v1.0) June 3, 2020



Table of Contents

Chapter 1: Introduction.....	4
Features.....	4
IP Facts.....	5
Chapter 2: Overview.....	6
Feature Summary.....	6
Unsupported Features.....	7
Licensing and Ordering.....	7
Chapter 3: Product Specification.....	8
Performance and Resource Utilization.....	9
Port Descriptions.....	9
Register Space.....	11
Chapter 4: Designing with the Core.....	13
Operation.....	13
Decoupling Using Global Routing.....	20
Clock Domains.....	20
Use Cases.....	24
Finding a Suitable Interface.....	30
Creating a Suitable Interface.....	31
Clocking.....	32
Resets.....	33
Chapter 5: Design Flow Steps.....	34
Customizing and Generating the Core.....	34
Constraining the Core.....	41
Simulation.....	42
Synthesis and Implementation.....	42
Chapter 6: Test Bench.....	43
Demonstration Test Bench.....	43

Appendix A: Upgrading	44
Appendix B: Debugging	46
Finding Help on Xilinx.com.....	46
Debug Tools.....	47
Appendix C: Additional Resources and Legal Notices	49
Xilinx Resources.....	49
Documentation Navigator and Design Hubs.....	49
References.....	50
Revision History.....	50
Please Read: Important Legal Notices.....	51

Introduction

The Xilinx[®] Dynamic Function eXchange Decoupler (DFX Decoupler) IP core can be used to provide a safe and managed boundary between the static logic and a Reconfigurable Partition during dynamic reconfiguration.

The core can be customized for the number of interfaces, type of interfaces, decoupling functionality, status, and control.

Features

- All Interface types registered in the Vivado[®] Design Suite are supported, including custom interfaces.
- Non-Vivado Design Suite interfaces are supported.
- Multiple interfaces per decoupler instance.
- The decoupling behavior can be configured for each interface.
- Each interface can have Clock Domain Crossing support.
- Optional Signal based control.
- Optional Signal based status.
- Optional AXI4-Stream based control.
- Optional AXI4-Stream based status.
- Optional AXI4-Lite based status and control.

IP Facts

LogiCORE™ IP Facts Table	
Core Specifics	
Supported Device Family ¹	UltraScale+™, UltraScale™, Zynq®-7000 SoC, 7 series
Supported User Interfaces	AXI4-Lite, AXI4-Stream
Resources	Performance and Resource Use web page
Provided with Core	
Design Files	Encrypted RTL
Example Design	Not Provided
Test Bench	VHDL
Constraints File	Xilinx Constraints File (XDC)
Simulation Model	Source HDL
Supported S/W Driver	N/A
Tested Design Flows²	
Design Entry	Vivado Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide .
Synthesis	Vivado Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 73351
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado® IP catalog.
2. For the supported versions of third-party tools, see the [Xilinx Design Tools: Release Notes Guide](#).

Overview

One or more DFX Decoupler cores can be used to make the interface between a Reconfigurable Partition and the static logic safe while dynamic reconfiguration is occurring. When active, user-selected signals crossing between the Reconfigurable Partition and the static logic are driven to user configurable values. When inactive, signals are passed unaltered.

Feature Summary

Interface Aware

The DFX Decoupler core is interface aware to make it faster to connect to standard interfaces.

Multiple Options for Status and Control

The DFX Decoupler core can be controlled and queried using single signals, an AXI4-Lite interface, or AXI4-Stream interfaces. Any combination of these interfaces can be enabled at the same time.

Clock Domain Crossing

The DFX Decoupler core inserts clock domain crossing synchronizers if requested to manage decoupled interfaces which run from a different clock than the core's control interface.

Works with Dynamic Function eXchange Controller

The DFX Decoupler core connects directly to the Dynamic Function eXchange Controller using the signal based control interface. More information about this core is available in the *Dynamic Function eXchange Controller IP LogiCORE IP Product Guide* ([PG374](#)).

Unsupported Features

The DFX Decoupler core should not normally be configured using the Vivado Design Suite `set_property` command. Instead, a custom `set_property` command is provided with the core. For more information, see [Configuring the Core Using Tcl Commands](#). The exceptions to this are bus interface parameters, such as reset polarity and clock `FREQ_HZ`, which can be set using `set_property`.

Related Information

[Configuring the Core Using Tcl Commands](#)

Licensing and Ordering

This Xilinx® LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the [Xilinx End User License](#).

For more information about this core, visit the [DFX Decoupler](#) product web page.

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

Designers of partially reconfigurable systems must consider several effects that can occur when reconfigurable modules are used.

- **Undesirable values driven into the static logic:** Signals driven by resources in the Reconfigurable Partition that is undergoing reconfiguration might take on undesirable values, namely:
 - They might glitch.
 - They might be driven to 1 by the interconnect which is undesirable if the signal is an active-High control signal.
 - They might be driven by a Reconfigurable Module that has not been reset and is therefore in an unknown state.
- **Corruption of the new Reconfigurable Module:** Signals driven by the static logic into the Reconfigurable Partition that is undergoing reconfiguration can cause the newly loaded Reconfigurable Module to become corrupted. For example:
 - Spurious writes to memories can occur.
 - Parts of the reconfigurable module can start to operate while other parts do not. This can occur even with RESET_AFTER_RECONFIG because the GWE (Global Write Enable) signal is asynchronous to the user clock, so it could write enable parts of the new Reconfigurable Module in one user clock cycle but leave other parts write disabled.

If any of these effects are problematic for a design, you can use decoupling logic to manage the signal values on the boundary between the static logic and the Reconfigurable Partition that is undergoing reconfiguration.

The signals that need to be decoupled are design-specific, and the following guidelines are recommended:

- Decouple all control signals generated from the Reconfigurable Partition.
- Decouple all control signals driven into the Reconfigurable Partition if the Reconfigurable Module being loaded cannot be fully reset before operation.
- If a Reconfigurable Module has logic that can start to execute without being qualified by a decoupled control signal, consider decoupling the appropriate input clock.

Performance and Resource Utilization

For full details about performance and resource use, visit the [Performance and Resource Use web page](#).

Port Descriptions

The port map of the DFX Decoupler core is dynamically created to match the configuration of the core. The following tables describe the ports.

The control and status ports are optional.

The Per-Interface Port Descriptions shows the ports for a single decoupled interface. Each DFX Decoupler core can decouple multiple interfaces.

Related Information

[Optional AXI4-Stream Status Channel Ports](#)
[Optional AXI4-Stream Control Channel Ports](#)
[Per-Interface Port Descriptions](#)

Clock and Reset Ports

Table 1: Clock and Reset Ports

Name	I/O	Description
aclk	I	Rising-edge clock used for the AXI4-Lite and AXI4-Stream interfaces.
s_axi_reg_aresetn	I	Synchronous active-Low reset for the AXI4-Lite interface.

Optional AXI4-Lite Register Ports

Table 2: Optional AXI4-Lite Register Interface Ports

Name	I/O	Description
s_axi_reg_awaddr	I	1-bit wide signal
s_axi_reg_awvalid	I	
s_axi_reg_awready	O	
s_axi_reg_wdata	I	32-bit wide signal. Only bit 0 is used.
s_axi_reg_wvalid	I	
s_axi_reg_wready	O	

Table 2: Optional AXI4-Lite Register Interface Ports (cont'd)

Name	I/O	Description
s_axi_reg_bresp	O	2-bit wide signal.
s_axi_reg_bvalid	O	
s_axi_reg_bready	I	
s_axi_reg_araddr	I	1-bit wide
s_axi_reg_arvalid	I	
s_axi_reg_arready	O	
s_axi_reg_rdata	O	32-bit wide signal. Only bit 0 is used. Other bits return 0.
s_axi_reg_rresp	O	2-bit wide signal.
s_axi_reg_rvalid	O	
s_axi_reg_rready	I	

Optional AXI4-Stream Status Channel Ports

Table 3: Optional AXI4-Stream Status Channel Ports

Name	I/O	Description
m_axis_status_tvalid	O	
m_axis_status_tdata	O	32-bit wide signal. Only bit 0 is used. Unused bits return 0.

Optional AXI4-Stream Control Channel Ports

Table 4: Optional AXI4-Stream Control Channel Ports

Name	I/O	Description
s_axis_ctrl_aresetn	I	Synchronous active-Low reset.
s_axis_ctrl_tvalid	I	
s_axis_ctrl_tready	O	
s_axis_ctrl_tdata	I	32-bit wide signal. Only bit 0 is used.

Decouple Interface Ports

Table 5: Decouple Interface Ports

Name	I/O	Description
decouple	I	Active-High control signal. When asserted, decoupling is enabled.
decouple_status	O	A single-bit signal giving the decoupled status of the core. Can be optionally enabled and disabled.
decouple_ref_clk	I	A reference clock for the decouple control signal. Automatically enabled when clock domain crossing is required and there are no AXI interfaces enabled. Disabled otherwise.

Per-Interface Port Descriptions

In the following table, <name> is a user-defined name for the interface being decoupled. For example, `rp_<name>_data` could be:

- `rp_fifo_data`, where <name> = `fifo`.
- `rp_counter_data`, where <name> = `counter`.

Table 6: Per-Interface Port Descriptions

Name	I/O	Description
<code>rp_<name>_<signal></code>	I or O	A signal <signal> in interface <name> that is to be attached to the Reconfigurable Partition.
<code>s_<name>_<signal></code>	I or O	A signal <signal> in interface <name> that is to be attached to the static logic
<code><name>_ref_clk</code>	I	A reference clock for interface <name>. Automatically enabled when clock domain crossing is required for this interface. Disabled otherwise.
<code><name>_decouple_status</code>	O	The decouple status for interface <name>. Automatically enabled when clock domain crossing is required for this interface. Disabled otherwise.

Register Space

The DFX Decoupler core register space is summarized in the following table.

Table 7: Register Address Space

Address Space Offset	Name	Description
00h	CONTROL	Control register
00h	STATUS	Status register

CONTROL Register

The CONTROL register is write only, and is mapped to the same address as the STATUS register.

Table 8: Control Register (Offset 00h)

Bits	Name	Description
31:1	Reserved	Reserved
0	CMD	1: Turn decoupling on 0: Turn decoupling off

STATUS Register

The STATUS register is read only and is mapped to the same address as the CONTROL register.

Table 9: Status Register (Offset 00h)

Bits	Meaning	Details
31:1	Reserved	Reserved
0	STATE	1: Decoupling is enabled 0: Decoupling is disabled

Designing with the Core

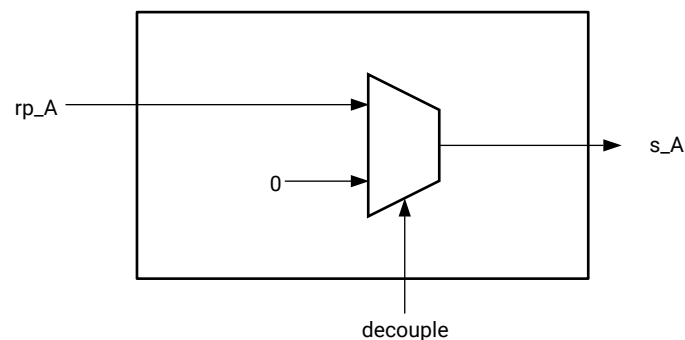
This section includes guidelines and additional information to facilitate designing with the core.

Operation

The following figure shows a simple decoupler that decouples a single signal.

- When the decouple input is 0, the `rp_A` input is passed unchanged to the `s_A` output.
- When the decouple input is 1, the `rp_A` input is decoupled from the `s_A` output, and `s_A` is driven to a constant value.

Figure 1: Simple DFX Decoupler Core



X14894-012420

Configuring a DFX Decoupler core to decouple the static from a Reconfigurable Partition can be time consuming when done one signal at a time, so the DFX Decoupler core has been designed to be interface aware.

According to the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994),

“An interface is a grouping of signals that share a common function. An AXI4-Lite master, for example, contains a large number of individual signals plus multiple buses, which are all required to make a connection. If each signal or bus is visible individually on an IP symbol, the symbol will be visually very complex. By grouping these signals and buses into an interface, the following advantages can be realized. First, a single connection in IP integrator (or Tcl command) will make a master to slave connection. Next, the graphical representation of this connection will be simple - a single connection. Finally, Design Rule Checks (DRCs) that are aware of the specific interface can be run to assure that all the required signals are connected properly.”

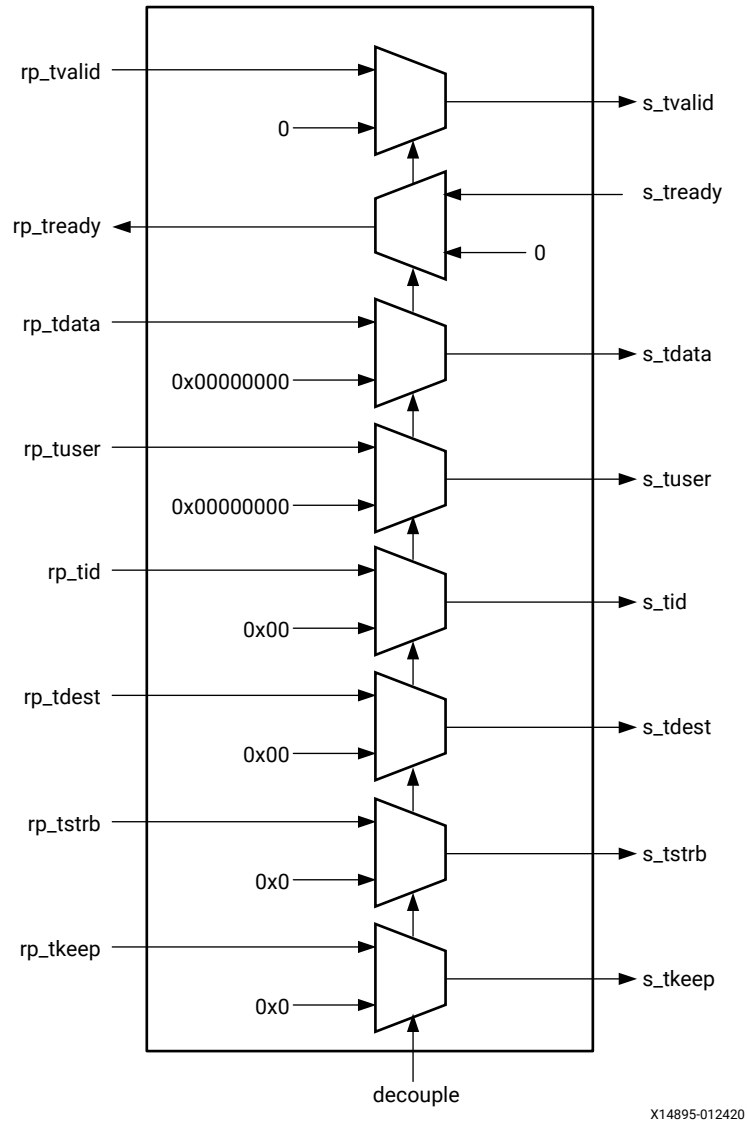
In the simplest case, an interface can contain one signal, such as `CLOCK`, `DATA`, or `RESET`.



TIP: To view the interfaces available in the Vivado® Design Suite, select **Window** → **IP Catalog**, and click the **Interfaces** tab.

By specifying a defined interface, the DFX Decoupler core can create the appropriate code for all signals in the interface. For example, the following figure shows a full decoupler when the AXI4-Stream interface is selected.

Figure 2: Fully Decoupled AXI4-Stream Interface



The decoupling behavior for each decoupled interface can be customized. With the exception of the AXI4 interfaces listed in the following table, all signals in each interface are present and decoupled to 0. Decoupling all signals in an interface can waste resources, so the common interfaces in the following table have different defaults.

Table 10: AXI4 Interfaces Decoupling Exceptions

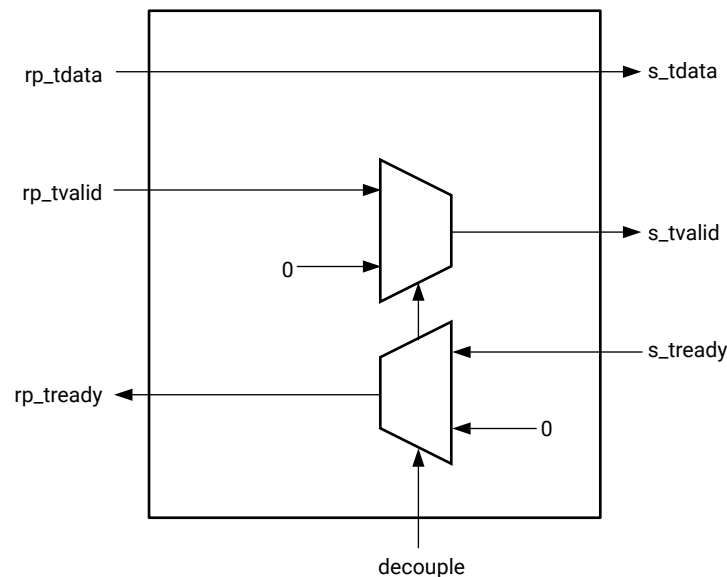
Interface	Signal Decoupling Behavior
AXI4-Stream	Only the TVALID and TREADY signals are decoupled. TID, TDEST, TSTRB, TUSER, and TKEEP signals are not present by default.

Table 10: AXI4 Interfaces Decoupling Exceptions (cont'd)

Interface	Signal Decoupling Behavior
AXI4-MM	Only the channel VALID and READY signals are decoupled. The channel USER, LAST, ID, LOCK, CACHE, PROT, REGION, and QoS signals are not present by default. Additionally, when the PROTOCOL parameter is set to: <ul style="list-style-type: none"> • AXI4LITE – the following signals are disabled: AWLEN, ARLEN, AWSIZE, ARSIZE, AWBURST, ARBURST, AWLOCK, ARLOCK, AWCACHE, ARCACHE, WLAST, and RLAST. • AXI3 – the following signals are fixed to 4 bits wide: AWLEN and ARLEN. • AXI4 – the following signals are fixed to 8 bits wide: AWLEN and ARLEN. • None – no restrictions are applied.

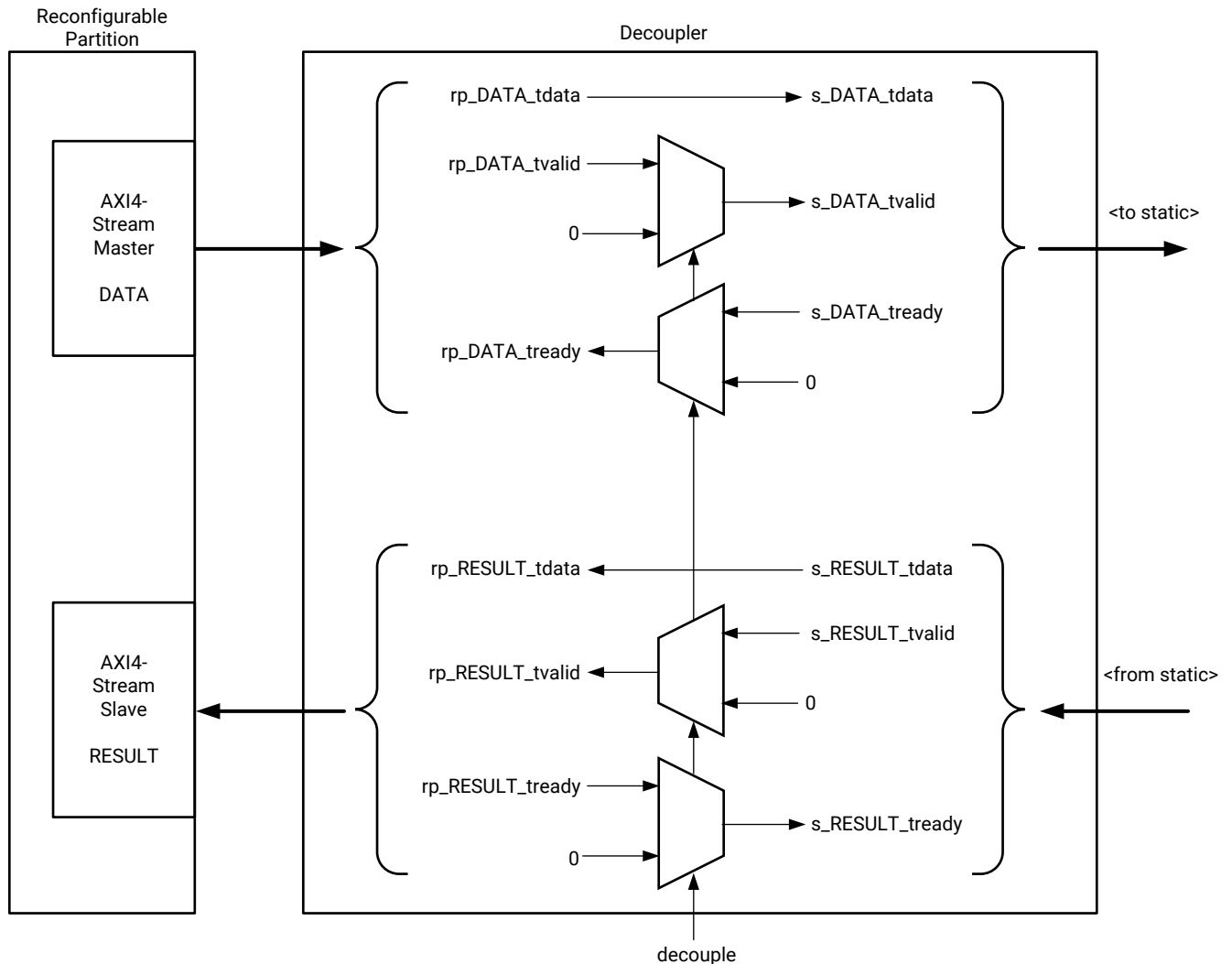
The following figure shows the default implementation for an AXI4-Stream decoupler.

Figure 3: Default Decoupling for AXI4-Stream Interfaces



The examples given so far have shown a decoupler that decouples a master interface on a Reconfigurable Partition. A master interface is one which initiates transactions, and a slave interface is one that responds to transactions. In the case of an interface with one signal, the master interface is the one that drives the signal, and the slave interface is the one that receives the signal. The DFX Decoupler core can also decouple slave interfaces on a Reconfigurable Partition, as shown in the following figure.

Figure 4: Decoupling a Master and a Slave AXI4-Stream Interface



X14897-012420

The previous figure also shows that one instance of a decoupler can decouple multiple Reconfigurable Partition interfaces. These can be of any interface type.

Control

The DFX Decoupler core provides several ways to control decoupling:

- Decouple signal:** The decouple signal input to the DFX Decoupler core is not registered, so it directly controls the multiplexors in the decoupler. Set to 1 to enable decoupling, and set to 0 to disable decoupling from this interface.
- AXI4-Lite interface:** The AXI4-Lite interface provides access to a register that holds the control value to use. A write of 1 to the control register enables decoupling, and a write of 0 to this register disables decoupling from this interface.

- AXI4-Stream interface:** The AXI4-Stream interface provides access to a register that holds the control value to use. A write of 1 to the control register enables decoupling, and a write of 0 to this register disables decoupling from this interface.

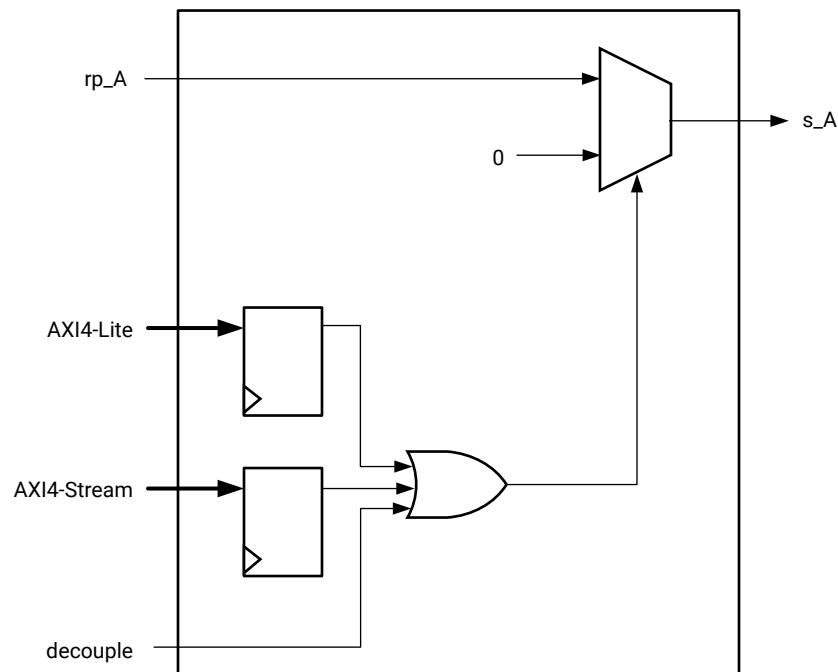
These can be enabled in any combination, as long as at least one is enabled (see the following table). The value used internally to control decoupling is the OR of all of these interfaces.

Table 11: Decoupling Control Combinations

AXI4-Stream Interface	AXI4-Lite Interface	Decouple Signal	Decoupling
0	0	0	Off
-	-	1	On
-	1	-	On
1	-	-	On

The following figure shows how these control interfaces combine to control the decoupling.

Figure 5: DFX Decoupler With All Control Options Enabled



X14898-012420

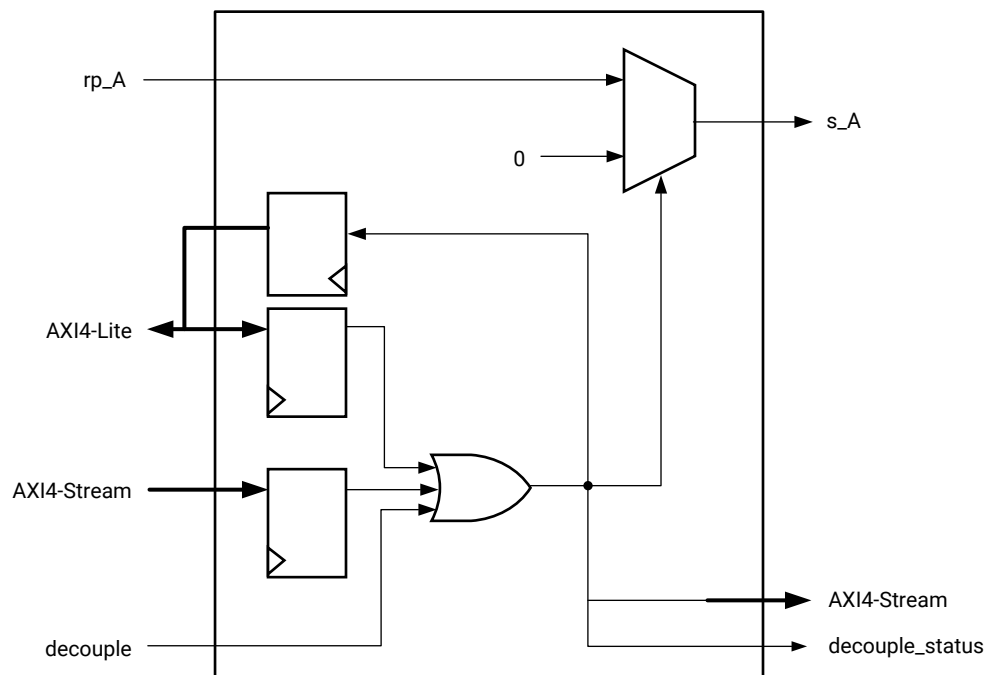
Status

The DFX Decoupler core provides several ways to get the status of the decoupling:

- **decouple_status signal:** The `decouple_status` signal output from the DFX Decoupler core is not registered, so it gives the immediate status of the decoupling. This can be used to control other DFX Decoupler cores.
- **AXI4-Lite:** The AXI4-Lite interface provides access to a register that holds the decoupling status.
- **AXI4-Stream interface:** The interface provides access to the real-time value of the decoupling status.

These can be enabled in any combination. You can have no status interfaces enabled. The following figure shows a DFX Decoupler core with all of the control and status interfaces enabled.

Figure 6: DFX Decoupler Core With All Control and Status Options Enabled



X14899-012420

Additionally, each interface with Clock Domain Crossing enabled automatically gets an output status signal called `<interface name>_decouple_status`. This shows the value of the decoupling status in the interface's clock domain.

Decoupling Using Global Routing

By default, all signals selected for decoupling are decoupled using a LUT inferred by the Vivado tools. If the signal being decoupled is either a clock or a signal that needs to remain on the global routing network, the decoupler can be configured to use a BUFGCE or a BUFGCTRL primitive to implement the multiplexing.

- The BUFGCE primitive provides glitch free decoupling, but decoupling only takes effect after the input signal switches to the decoupled value. The BUFGCE primitive should only be used for clocks.
- The BUFGCTRL primitive provides instant decoupling, but it is not guaranteed to be glitch free because the IGNORE0 input is tied to 1. For more information, see the appropriate clocking guide: *7 Series FPGAs Clocking Resources User Guide (UG472)*, or *UltraScale Architecture Clocking Resources User Guide (UG572)*.

If location constraints are required for these primitives, the instance name is constructed as follows:

```
b_<interface name>.<interface name>_<signal name>_<bit>_bufgce  
b_<interface name>.<interface name>_<signal name>_<bit>_bufgctrl
```

In the case of single bit signals, <bit> is 0.

For example, for an interface called `intf_0` that contained a clock called `CLK`, the name would be:

```
b_intf_0_intf_0_CLK_0_bufgce
```

Clock Domains

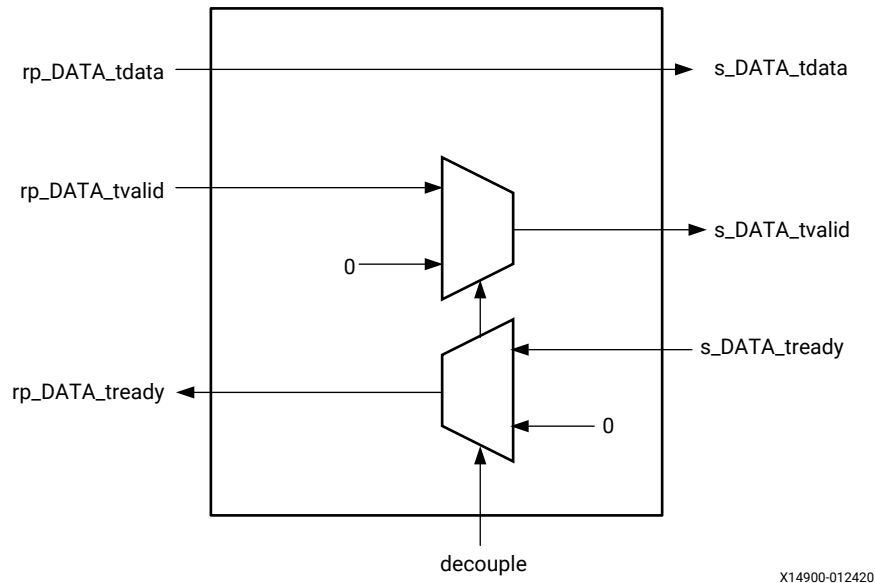
If the clock used to generate signals for a decoupled interface is different from the clock that manages the decoupler's control interfaces, clock domain crossing can be enabled for that interface. When clock domain crossing is enabled, a reference clock input for the decoupled interface is added to the core's port map. This provides the clock to use for the secondary stages of the clock domain crossing synchronizer.

The clock for the primary stage of the clock domain crossing synchronizer is the AXI clock if available. If the AXI clock is not present (because all of the AXI4-Stream and AXI4-Lite interfaces are disabled), a reference clock called `decouple_ref_clk` is enabled on the port map, and that is used to clock the primary stage of the clock domain crossing synchronizer.

Example 1

The following figure shows a decoupler for an AXI4-Slave interface which requires no clock domain crossing.

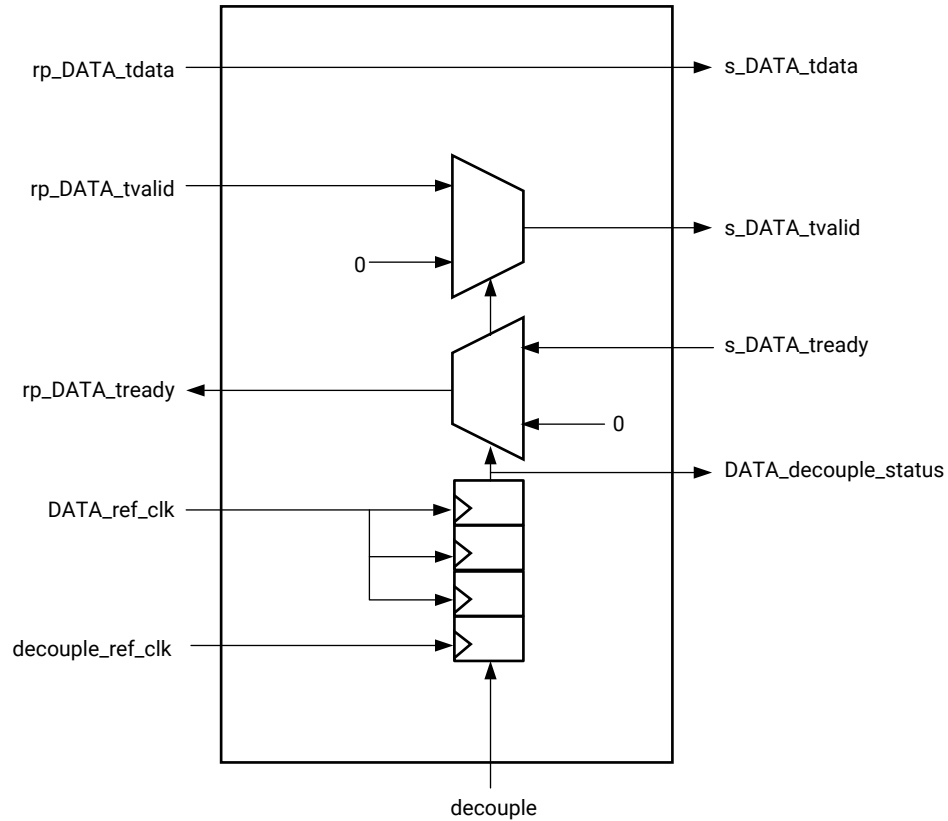
Figure 7: Example 1 With No Clock Domain Crossing



X14900-012420

The following figure shows the same example, but with a three-stage clock domain crossing synchronizer. Because there is no AXI clock in this example, a reference clock for the control interface is automatically included.

Figure 8: Example 1 With Three Stages of Clock Domain Crossing

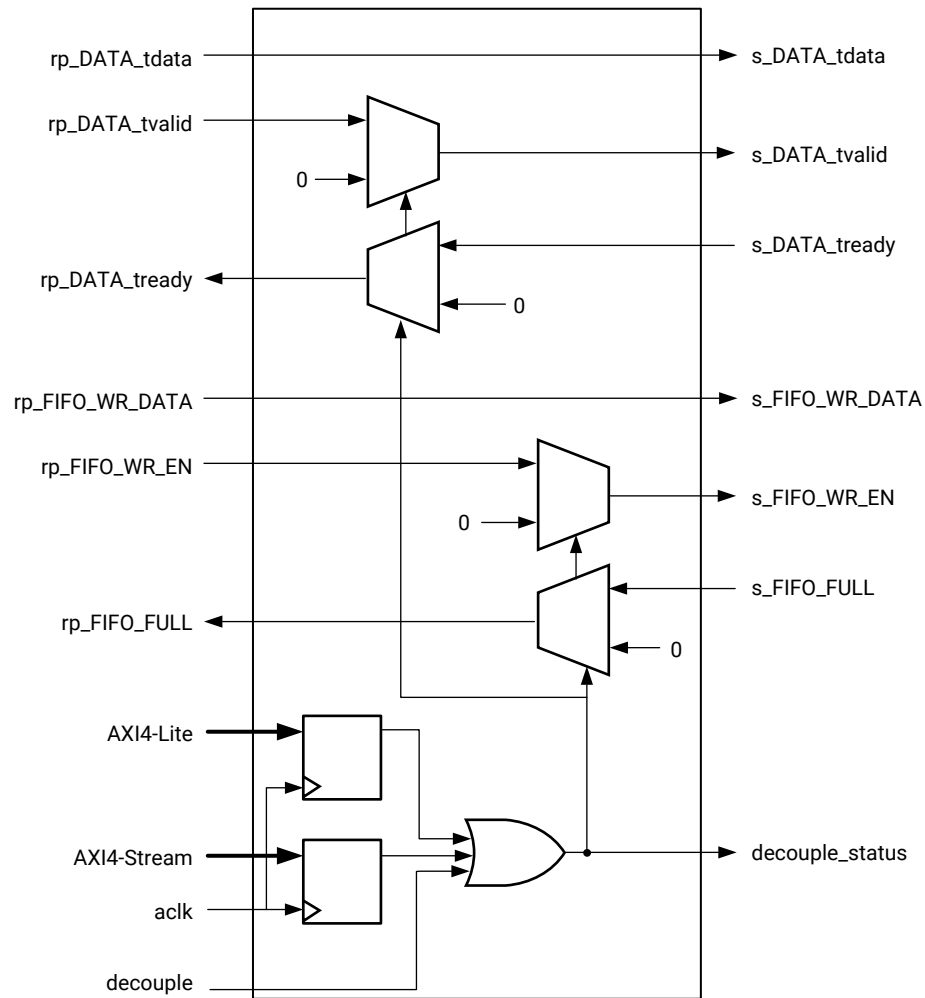


X14901-012420

Example 2

The following figure shows a more complex decoupler without clock domain crossing. This decoupler handles an AXI4-Stream interface called `DATA` and a FIFO Write channel interface called `FIFO`.

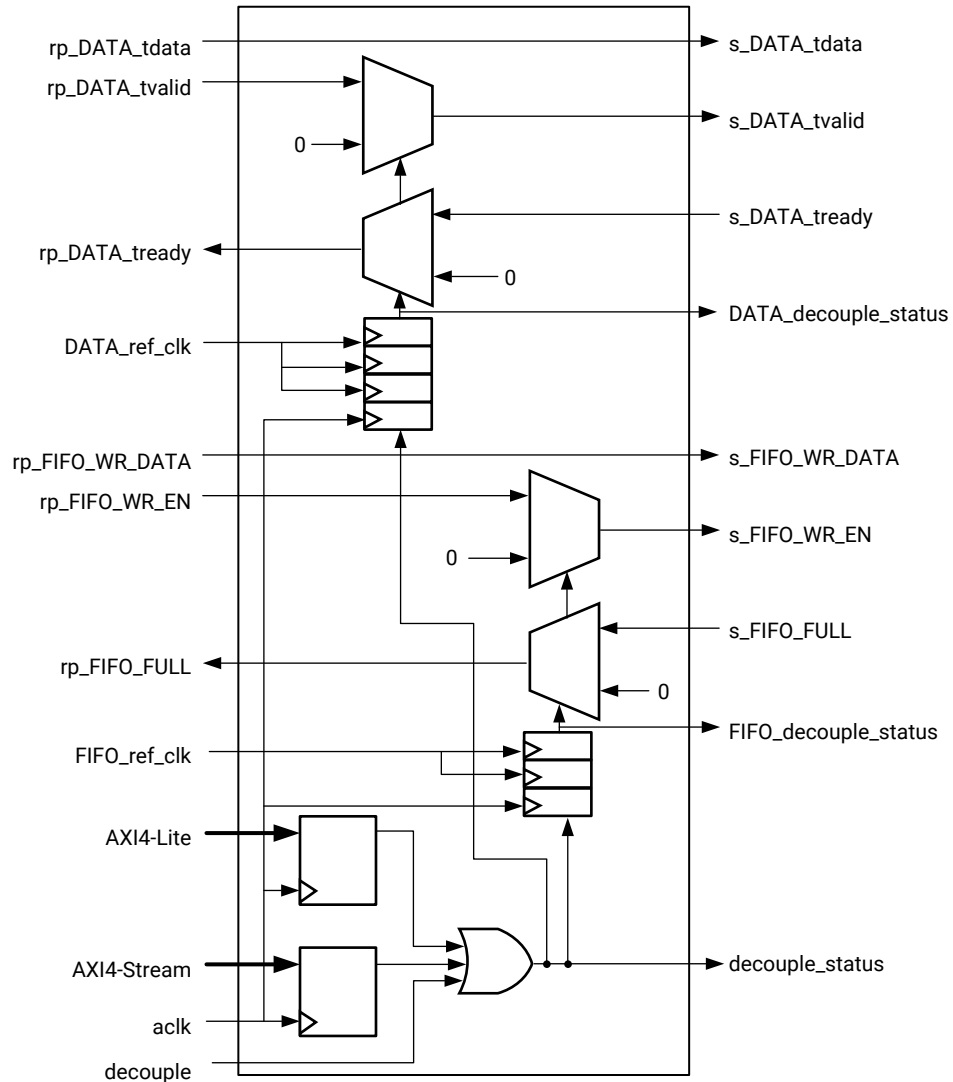
Figure 9: Example 2 With No Clock Domain Crossing



X14902-012420

The following figure shows the same example, but with a three-stage clock domain crossing synchronizer on the DATA interface and a two-stage clock domain crossing synchronizer on the FIFO interface. Both of these interfaces have their own reference clock (DATA_ref_clk and FIFO_ref_clk) but decouple_ref_clk is not needed in this example because there is already an AXI clock for the control interfaces.

Figure 10: Example 2 With Clock Domain Crossing Enabled



X14903-012420

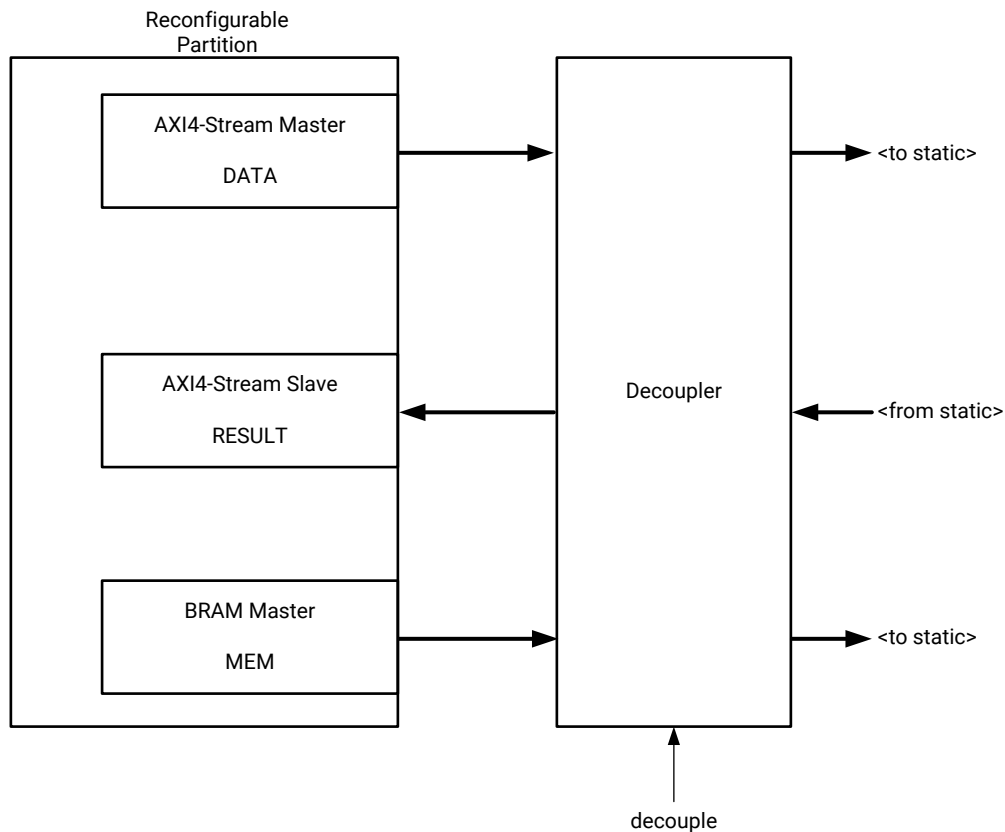
Use Cases

This section shows various use cases for the DFX Decoupler core.

One DFX Decoupler Per Reconfigurable Partition

In the use case shown in the following figure, one DFX Decoupler instance is used to manage all of the interfaces that exist between a Reconfigurable Partition and the static logic.

Figure 11: One DFX Decoupler Core Per Reconfigurable Partition



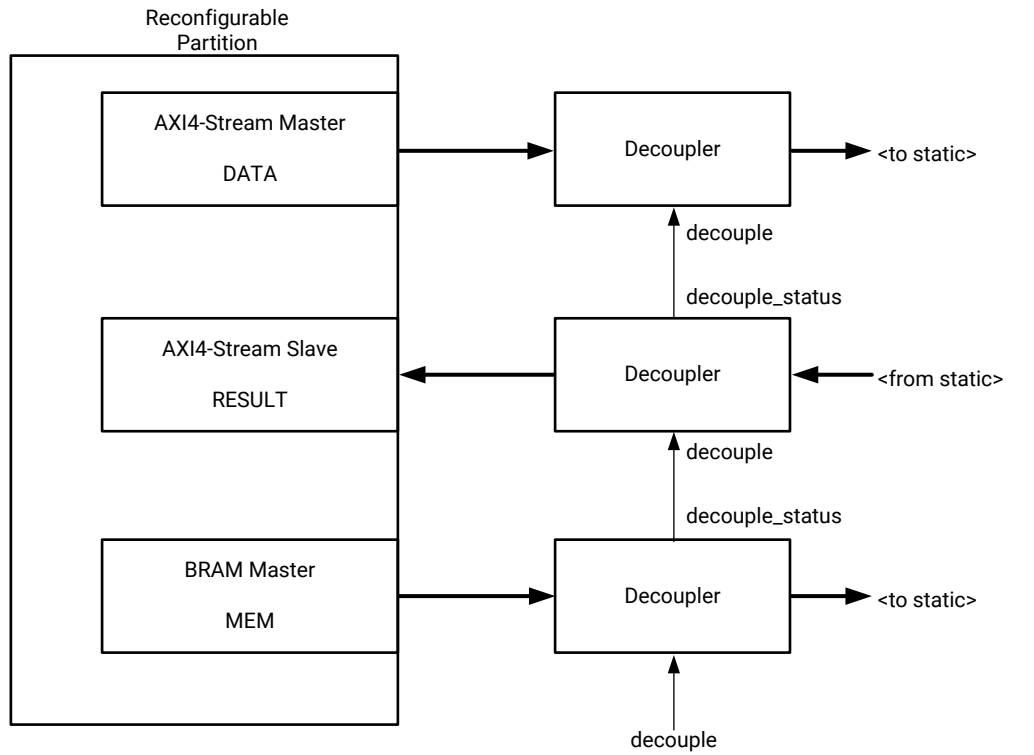
X14904-012420

Multiple DFX Decoupler Cores per Reconfigurable Partition

There is no need to have only one DFX Decoupler instance per Reconfigurable Partition. It might be easier to use multiple decouplers. For example, it might be desirable to have:

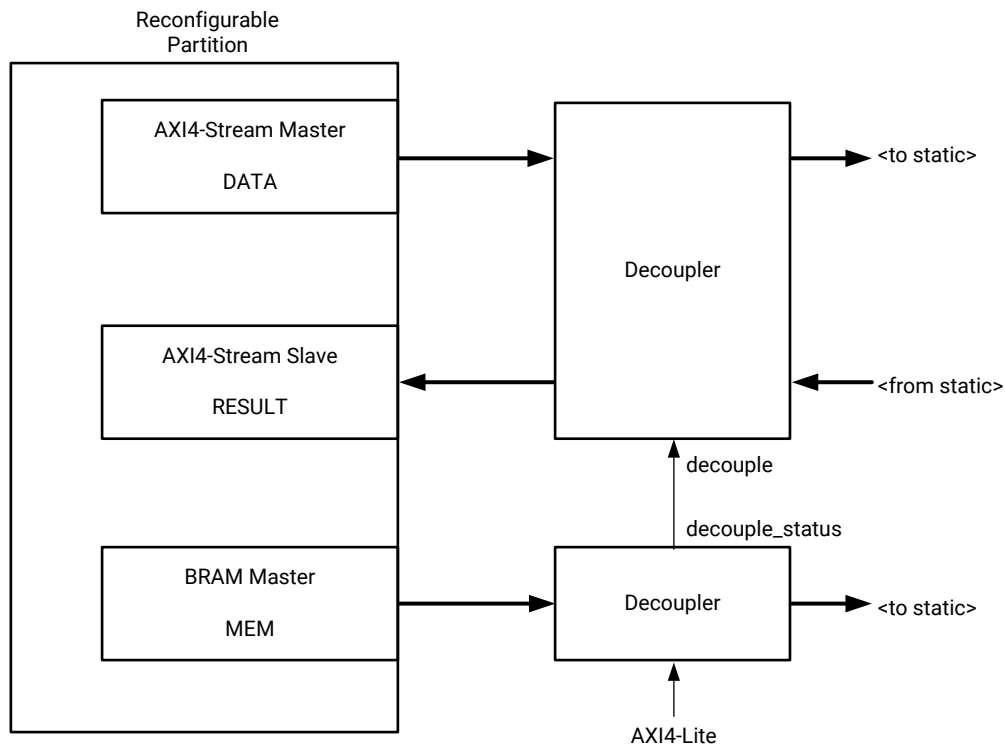
1. One decoupler per interface (see the following figure).
2. One decoupler for all master interfaces, and one for all slave interfaces.
3. One decoupler of each interface type, regardless of interface mode (see [Figure 13: One DFX Decoupler Core Per Interface Type](#)).

Figure 12: One DFX Decoupler Core Per Interface



X14905-012420

Figure 13: One DFX Decoupler Core Per Interface Type



X14906-012420

Other use cases are possible as well. The exact use case you use might be influenced by such things as:

- A desire to reduce the number of IP instances in a design.
- The ease of layout in the Vivado IP integrator.
- The ease of declaration and connection in HDL-based designs.
- The speed and ease of configuring many small decoupler instances, versus one large decoupler.
- Whether phased decoupling is required (see Phased Decoupling).

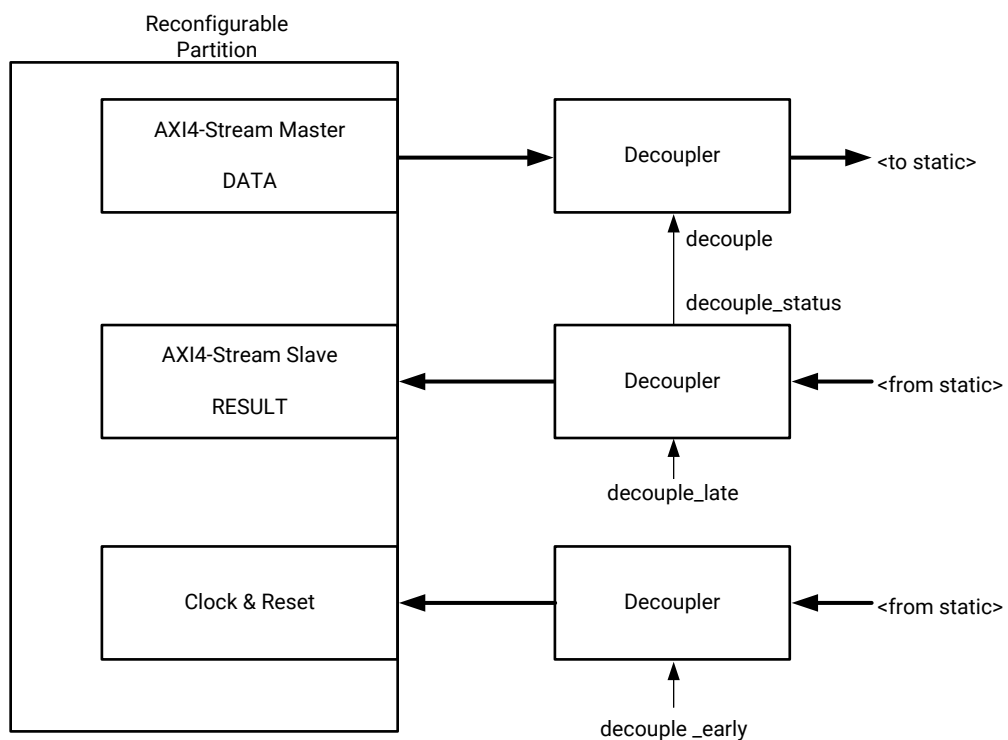
When multiple decouplers are used per Reconfigurable Partition, one can be designated as a master and the rest as slaves. The master decoupler is controlled by you, and the slaves are controlled by the `decouple_status` output of the master decoupler or of other slaves.

[Figure 12: One DFX Decoupler Core Per Interface](#) shows the bottom decoupler as the master decoupler, and in this case each slave decoupler is controlled by the decoupler next to it. The master decoupler can use any control interface type that is appropriate to the design. The previous figure shows an AXI4-Lite interface being used.

Phased Decoupling

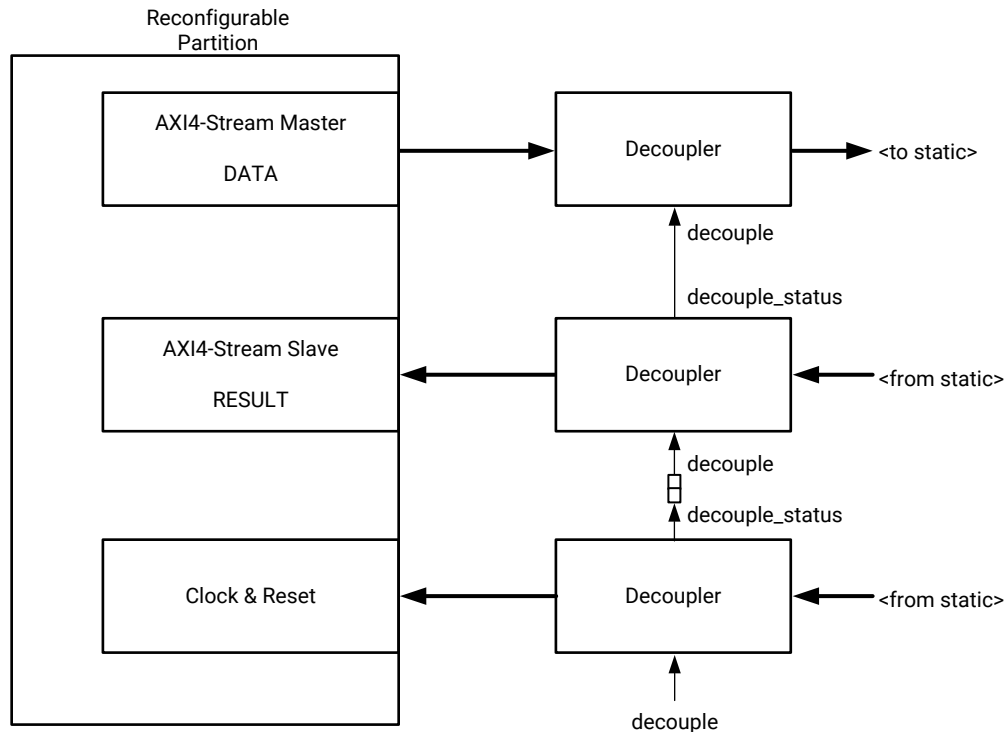
You might need to decouple/recouple different interfaces in a Reconfigurable Partition at different times. An example of this is to recouple the clock and reset inputs to a Reconfigurable Partition so that the Reconfigurable Module can be put into reset before the other interfaces are recoupled. The following figure shows this being handled with two independently controlled signals (`decouple_early` and `decouple_late`). [Figure 15: Phased Decoupling With One Control Signal and a Delay Line](#) shows this with one control signal and a two-clock cycle delay between the master decoupler and the slave decouplers.

Figure 14: Phased Decoupling With Independent Control Signals



X14907-012420

Figure 15: Phased Decoupling With One Control Signal and a Delay Line

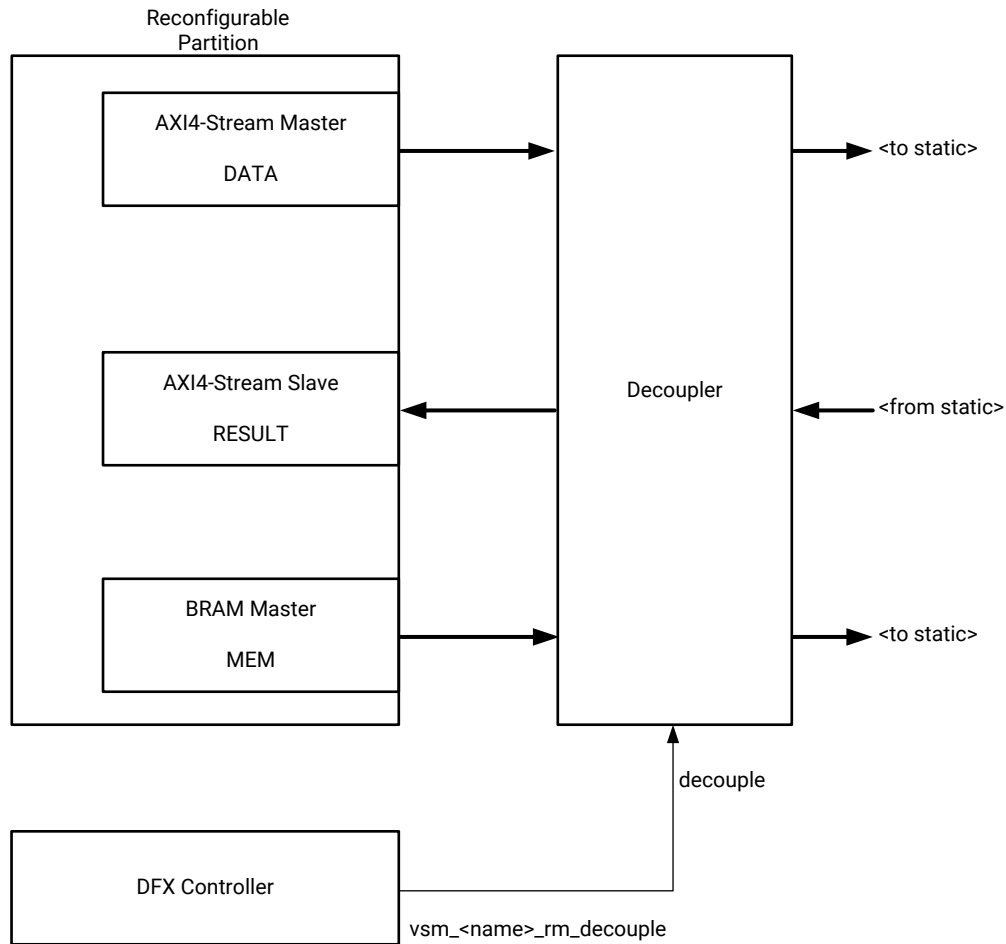


X14908-012420

Integration with the Dynamic Function eXchange Controller

The DFX Decoupler core has been designed to operate with the Dynamic Function eXchange Controller core, as shown in the following figure. The `rm_decouple` output of the appropriate Virtual Socket Manager can be connected directly to the `decouple` control signal of the DFX Decoupler core. If multiple decouplers are used, it can be connected to each `decouple` control signal directly, or the decouplers can be configured as shown in the master and slave use case, as previously discussed.

Figure 16: Dynamic Function eXchange Decoupler Core Controlled by the Dynamic Function eXchange Controller



X14909-020320

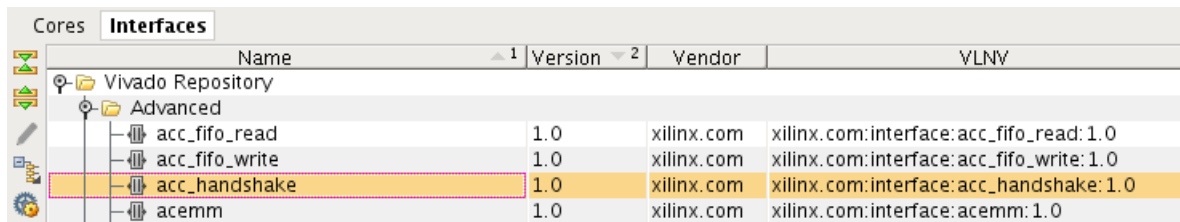
Finding a Suitable Interface

The Vivado Design Suite provides many pre-defined interfaces which can be viewed from the Interface Catalog window in the Vivado Integrated Design Environment (IDE) (see the following figure).



TIP: To open the Interface Catalog window, select **Window** → **IP Catalog** and click the **Interfaces** tab.

Figure 17: Interface Catalog Window



Name	Version	Vendor	VLN
Vivado Repository			
Advanced			
acc_fifo_read	1.0	xilinx.com	xilinx.com:interface:acc_fifo_read:1.0
acc_fifo_write	1.0	xilinx.com	xilinx.com:interface:acc_fifo_write:1.0
acc_handshake	1.0	xilinx.com	xilinx.com:interface:acc_handshake:1.0
acemm	1.0	xilinx.com	xilinx.com:interface:acemm:1.0

Double-click an interface in the Interface Catalog to open a window that contains more information about the interface as in the following figure.

Figure 18: Interface Details Pane



Details	
Name:	acc_handshake
Version:	1.0
Description:	Accelerator Handshake Interface
Vendor:	xilinx.com
VLN:	xilinx.com:interface:acc_handshake:1.0
Extends VLN:	
Max Masters:	1
Max Slaves:	1

Creating a Suitable Interface

If none of the existing interface types are suitable for your application, you have two options for creating a suitable interface.

The first option is to create an interface definition in the Vivado IDE by selecting **Tools** → **Create Interface Definition**. This is a useful approach if you need to use the interface in multiple decouplers.

The second option is to create an ad-hoc interface in the decoupler using the DFX Decoupler core customization page or configuration properties. To do this, create an interface of type `undef` and add signals that you require, as shown in the following figure.

Figure 19: Creating an Custom Undef Interface

Interface Options

Interface to Configure:

Interface Name:

Interface VLNV:

The Protocol overlay to apply to the VLNV:

Interface Mode at RP:

Number of CDC Stages:

Signal Options

New signal name:

Name	Present	Delete	Direction at the RP
signal 1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	As defined in the inter
No Signal	<input type="checkbox"/>	<input type="checkbox"/>	As defined in the inter

Clocking

There are four clocking scenarios possible with the core:

1. The instance has no clocks.
2. The instance has a single AXI clock.
3. The instance has a single AXI clock and one or more interface reference clocks.
4. The instance has a single reference clock for the decouple control signal, and one or more interface reference clocks.

Scenarios 3 and 4 occur when the clock used to drive the control interface(s) is different from the clock used to generate signals to be decoupled. In this case, clock domain crossing hardware is enabled and reference clocks are required. These scenarios are discussed in detail in Clock Domains.

Each clock on the core interface, whether it is a control clock, a reference clock, or a clock to be decoupled, has a parameter `FREQ_HZ` which can be used to set the clock frequency. When the core is generated in out-of-context mode, these frequencies are used to set clock constraints on the core instance. These frequencies are also used in IP integrator for error checking.

Note: A clock is a signal within an interface of type `xilinx.com:signal:clock_rtl`. No other signal gets a `FREQ_HZ` parameter, even if it is named to look like a clock.

The property names are defined as:

```
CONFIG.<clock name>.FREQ_HZ
```

where `<clock name>` is one of:

- `aclk_CLOCK` for the AXI control interface clock.
- `decouple_ref_clk_CLOCK` for the decouple signal interface reference clock.

Some examples are found in the following table.

Table 12: Property Names Examples

Property Name	Description
CONFIG.aclk_CLOCK.FREQ_HZ	The AXI control interface clock.
CONFIG.intf_0_CLK_CLOCK.FREQ_HZ	A clock called CLK in an interface of type <code>xilinx.com:signal:clock_rtl:1.0</code> called <code>intf_0</code> .

These properties can be set using the standard Vivado `set_property` command. Additional constraints might be required when decoupling clocks using a `BUFGCE` primitive or a `BUFGCTRL` primitive. The Vivado tools treat the data inputs to these primitives as clocks, so the relationship between the data input and its control input (the decouple signal) might need to be defined using constraints.

Related Information

[Clock Domains](#)

Resets

The core has two reset signals:

- `s_axi_reg_aresetn`: An active-Low reset for the AXI4-Lite interface.
- `s_axis_ctrl_aresetn`: An active-Low reset for the AXI4-Stream control interface.

These signals are only present on the core boundary when the appropriate interfaces are enabled. They only need to be asserted for a single clock cycle to take effect.

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.

Validation Tab

This tab is available on the left of the core customization page. The Validation tab lists any errors that remain with the current configuration. The core cannot be generated until all listed errors are fixed. When the configuration contains no errors, the tab displays the text "There are no errors."

Information Tab

This tab is available on the left of the core customization page. The Information tab provides the following information:

- The number of BUFGCTRL primitives used by the core instance. Because BUFGCE primitives are just BUFGCTRL primitives configured in a certain way, they are included in this count.
- The currently selected interface type.
- The signals that are enabled in the interface and if they are decoupled.
- The signals that are disabled in the interface

Global Options

This tab is available on the right of the core customization page. The Global Options tab is used to configure the parts of the core that do not depend on the number of interfaces, or their configuration. The following options are available:

- **Enable the AXI4-Lite Interface:** Enables or disables the AXI4-Lite register interface.
- **Enable the AXI4-Stream Control Channel:** Enables or disables the AXI4-Stream control channel.
- **Enable the AXI4-Stream Status Channel:** Enables or disables the AXI4-Stream status channel.
- **Enable the Decouple control signal:** Enables or disables the decouple control signal.
- **Enable the Decouple status signal:** Enables or disables the decouple status signal.

Interface Options

This tab is available on the right of the core customization page. The Interface Options tab is used to configure the interfaces that the DFX Decoupler core manages.

The tab is split into three areas which are discussed separately:

1. Control Buttons
2. Interface Options

3. Signal Options

The screenshot shows the 'Global Options' dialog box with the 'Interface Options' tab selected. It is divided into three numbered sections:

- Control Buttons:** Contains 'New Interface' and 'Delete Interface' buttons.
- Interface Options:** Contains several configuration fields:
 - Interface to Configure: `intf_0 (xilinx.com:interface:aximm rti:1.0)`
 - Interface Name: `intf_0`
 - Interface VLNV: `xilinx.com:interface:aximm rti:1.0`
 - The Protocol overlay to apply to the VLNV: `AXI4`
 - Interface Mode at RP: `Master`
 - Number of CDC Stages: `0`
- Signal Options:** Contains a 'New signal name' text box and a table of signals.

Name	Present	Direction at the RP	Width (bits)	Decouple	Decouple Value (hex)	Resource Type
ARVALID	<input checked="" type="checkbox"/>	As defined in the interface specification	1	<input checked="" type="checkbox"/>	0x0	Infer
ARREADY	<input checked="" type="checkbox"/>	As defined in the interface specification	1	<input checked="" type="checkbox"/>	0x0	Infer
AWVALID	<input checked="" type="checkbox"/>	As defined in the interface specification	1	<input checked="" type="checkbox"/>	0x0	Infer

Control Buttons

There are two control buttons available, one of which is automatically hidden when not required:

- **New Interface:** Click this button to add a new interface to the core.
- **Delete Interface:** Click this button to delete the currently selected interface. This button is only available when an interface is selected.

Interface Options

- **Interface to Configure:** This drop-down list contains the names of all the interfaces in the core. Select the interface you want to configure.
- **Interface Name:** This option contains the name of the selected interface. You can change the text in this box to change the name of the interface. The name must satisfy the following rules:
 - Contains only letters, numbers or "_" (underscore).
 - Does not start or end with "_" (underscore).
 - Does not contain "__" (double underscore).

Note: The name change only takes effect when you click another control in the customization page.

- **Interface VLNV:** Select the interface vendor, library, name, and version (VLNV) definition to use for this interface. Selecting undef lets you define the interface signals directly.

- **Protocol Overlay to Apply to the VLNV:** Select the protocol to apply to this VLNV. Protocols apply some restrictions, such as presence and width, to some of the signals in the interface. This is only enabled for AXI4-MM interfaces and is only required to enable the Vivado IP integrator to connect the AXI-MM interfaces correctly.
- **Interface Mode at Reconfigurable Partition:** Select the mode of the interface (master or slave) at the reconfigurable partition boundary.
- **CDC Stages:** The number of synchronization stages to use when crossing between clock domains. Valid values are 0, 2, 3, 4, 5, 6.

Related Information

[Operation](#)

Signal Options

- **New Signal Name:** Enter the name of the new signal to create. This option is only available when the Interface Type is undef. The name must satisfy the following rules:
 - Contains only letters, numbers or "_" (underscore).
 - Does not start or end with "_" (underscore).
 - Does not contain "__" (double underscore).

Note: The name change only takes effect when you click another control in the customization page.

- **Name:** [Read only] The name of the signal in each row of the table.
- **Management:** [In IP integrator only] Select between user managed (manual) and IP integrator managed (auto).
- **Present:** Check the box to include the signal in the decoupler's port map. Uncheck the box to not include the signal in the decoupler's port map.
- **Delete:** [Undef interfaces only] Check the box to delete this signal.
- **Note:** This triggers the delete so the check mark never actually appears in the box.
- **Direction:** Chose the direction of the signal. The options are:
 - **Input to the Reconfigurable Partition:** Uses the signal as an input to the Reconfigurable Partition regardless of the interface mode.
 - **Output from the Reconfigurable Partition:** Uses the signal as an output from the Reconfigurable Partition regardless of the interface mode.
 - **As specified in the interface definition:** Use the direction specified in the interface definition.
 - **Reversed from the interface definition:** Use the reverse of the direction specified in the interface definition.

- **Width:** The width of the signal in bits.
- **Decouple:** Check this box to decouple the signal in Decouple mode. Uncheck this box if the signal should pass through in Decouple Mode.
- **Decouple Value (hex):** The signal is decoupled to this value. This field is not verified against the width of the signal. The MSBs in the value that cannot map to bits in the signal are quietly ignored.
- **Resource Type:** How the decoupling should be implemented for this signal. The options are:
 - **Buffer:** Use BUFGCTRL primitives to decouple each bit of this signal. This is not glitch free but it allows decoupling to happen instantly.
 - **Clock Buffer:** Use BUFGCE primitives to decouple each bit of this signal. This is glitch free but the signal only becomes decoupled when the input itself switches to the decoupled value.
 - **Infer:** Let the tools infer a primitive to use. This is typically a LUT regardless of signal type or load.

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 13: User Parameters

Vivado IDE Parameter/Value ¹	User Parameter/Value ¹	Default Value
Enable the AXI4-Lite Interface	HAS_AXI_LITE	0
Enable the AXI4-Stream Control Channel	HAS_AXIS_CONTROL	0
Enable the AXI4-Stream Status Channel	HAS_AXIS_STATUS	0
Enable the Decouple Control Signal	HAS_SIGNAL_CONTROL	1
Enable the Decouple Status Signal	HAS_SIGNAL_STATUS	1
Interface to Configure	No equivalent User Parameter	
Interface Name	No equivalent User Parameter. The name of each Interface is specified as part of the user parameter name (shown as <i_name> below)	
Interface VLNV	INTF.<i_name>.VLNV Note: This parameter is case sensitive.	undef
Protocol to apply to the VLNV	INTF.<i_name>.PROTOCOL	AXI4 if the VLNV is AXI4; None otherwise
Interface Mode at Reconfigurable Partition	INTF.<i_name>.MODE	master
Master	master	
Slave	slave	

Table 13: User Parameters (cont'd)

Vivado IDE Parameter/Value ¹	User Parameter/Value ¹	Default Value
Number of CDC Stages	INTF.<i_name>.CDC_STAGES (0, 2, 3, 4, 5, 6)	0
New Signal Name	No equivalent User Parameter. The name of each signal is specified as part of the user parameter name (shown as <s_name> below)	
Name (in Table)	No equivalent User Parameter. The name of each signal is specified as part of the user parameter name (shown as <s_name> below)	
Present (in Table)	INTF.<i_name>.SIGNAL.<s_name>.PRESENT	Variable
Direction at the Reconfigurable Partition (in Table)	INTF.<i_name>.SIGNAL.<s_name>. DIRECTION	s
An input to the Reconfigurable Partition regardless of the interface direction	in	
An output from the Reconfigurable Partition regardless of the interface direction	out	
As defined in the interface specification	s (short for "same")	
Reversed from the interface specification	r (short for "reversed")	
Width (in Table)	INTF.<i_name>.SIGNAL.<s_name>. WIDTH	Variable
Decouple (in Table)	INTF.<i_name>.SIGNAL.<s_name>. DECOUPLED (0, 1)	Variable
Decouple Value (in Table)	INTF.<i_name>.SIGNAL.<s_name>. DECOUPLED_VALUE	Variable
Resource Type (in Table)	INTF.<i_name>.SIGNAL.<s_name>. RESOURCE	infer
Buffer	buffer	
Clock Buffer	clk_buffer	
Infer	infer	
Management (in Table). This is available in IP integrator only	INTF.<i_name>.SIGNAL.<s_name>. MANAGEMENT	auto
Auto	auto	
Manual	manual	

Notes:

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

Configuring the Core Using Tcl Commands

The DFX Decoupler core can be configured from the Tcl command line by setting properties directly. A custom `set_property` command is generally required. The following command must be executed in the Vivado Tcl command line to access

```
dfx_decoupler_v1_0::set_property:
```

```
source [get_property REPOSITORY \
      [get_ipdefs
      *dfx_decoupler:1.0]]/xilinx/dfx_decoupler_v1_0/tcl/api.tcl -notrace
```

Note: This can be placed in the Vivado Design Suite initialization file for ease of use.

Example 1: Decouple an AXI4-Stream Master on the Reconfigurable Partition

```
create_ip -vlnv xilinx.com:ip:dfx_decoupler:1.0 -module_name dfxd_0
dfx_decoupler_v1_0::set_property -dict [list \
  CONFIG.INTF.my_intf.VLNV xilinx.com:interface:axis_rtl:1.0 \
  ] [get_ips dfxd_0]
```

Example 2: Decouple an AXI4-Stream Master on the Reconfigurable Partition and an AXI4-MM slave on the Reconfigurable Partition

```
create_ip -vlnv xilinx.com:ip:dfx_decoupler:1.0 -module_name dfxd_1
dfx_decoupler_v1_0::set_property -dict [list \
  CONFIG.INTF.intf_0.VLNV xilinx.com:interface:axis_rtl:1.0 \
  CONFIG.INTF.intf_1.VLNV xilinx.com:interface:aximm_rtl:1.0 \
  CONFIG.INTF.intf_1.MODE slave \
  ] [get_ips dfxd_1]
```

Example 3: Decouple an AXI4-Stream Slave on the Reconfigurable Partition with TUSER Enabled and Decoupled

```
create_ip -vlnv xilinx.com:ip:dfx_decoupler:1.0 -module_name dfxd_2
dfx_decoupler_v1_0::set_property -dict [list \
  CONFIG.INTF.intf_0.VLNV xilinx.com:interface:axis_rtl:1.0 \
  CONFIG.INTF.intf_0.MODE slave \
  CONFIG.INTF.intf_0.SIGNAL.TUSER.PRESENT 1 \
  CONFIG.INTF.intf_0.SIGNAL.TUSER.WIDTH 32 \
  CONFIG.INTF.intf_0.SIGNAL.TUSER.DECOUPLED 1 \
  CONFIG.INTF.intf_0.SIGNAL.TUSER.DECOUPLED_VALUE 0xFFFFFFFF \
  ] [get_ips dfxd_2]
```

Interface parameters such as a reset polarity (POLARITY) or a clock frequency (FREQ_HZ) should be set using `set_property` as normal.

Output Generation

The DFX Decoupler core delivers Synthesis and Simulation models as standard. For details, see the *Vivado Design Suite User Guide: Designing with IP (UG896)*.

In addition, the core delivers a configuration information text file containing useful information about the core configuration. This can be accessed in the IP Sources tab of the Vivado IDE in the instances synthesis group.

The text file can also be found on disk at:

```
<ip source dir>/documentation/configuration_information.txt
```

To access the text file on disk, disable IP Core Containers, or extract the file using the `extract_files` command.

This text file contains the following information:

- The property values used to configure the core.
- A command to regenerate the core as configured.

Constraining the Core

The DFX Decoupler core does not normally require user constraints. However, some constraints might be required in certain circumstances, as detailed in this section.

Required Constraints

This section is not applicable for this IP core.

Device, Package, and Speed Grade Selections

The DFX Decoupler core works with any device supported by the Vivado Design Suite.

Clock Frequencies

When the core is used in an out-of-context mode, a constraints file is automatically generated to constrain all relevant clocks to the frequencies specified using the instance's `FREQ_HZ` properties. You set these properties as described in [Clocking](#).

When a signal is decoupled using a `BUFGCE` or a `BUFGCTRL`, additional constraints might be required by the tools, depending on how the instance is configured and how it is used in the system. Missing constraints will be highlighted by warning messages during implementation. For more details, see [Clocking](#).

Clock Management

This section is not applicable for this IP core.

Clock Placement

When a signal is decoupled using a BUFGCE or a BUFGCTRL, additional constraints might be desirable to specify exactly which resources are used. For information on the instance names of these primitives, see [Decoupling Using Global Routing](#).

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.

Related Information

[Clocking](#)

[Decoupling Using Global Routing](#)

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)*.



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

Synthesis and Implementation

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

Test Bench

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

Demonstration Test Bench

When the core is generated using the Vivado Design Suite, a demonstration test bench is created. This is a simple VHDL test bench that exercises the core.

The demonstration test bench source code is one VHDL file: `demo_tb/tb_<component_name>.vhd` in the Vivado output directory. The source code is comprehensively commented.

Using the Demonstration Test Bench

The demonstration test bench instantiates the generated core. Compile the netlist and the demonstration test bench into the work library (see your simulator documentation for more information on how to do this). Then, simulate the demonstration test bench. View the test bench signals in your simulator waveform viewer to see the operations of the test bench.

Upgrading

The DFX Decoupler IP core supersedes the Partial Reconfiguration Decoupler IP core. This section identifies any required migration changes.

Upgrading from the Partial Reconfiguration Decoupler to the DFX Decoupler

The DFX Decoupler IP core is a direct replacement for the Partial Reconfiguration Decoupler IP core and is functionally equivalent. When adding a Partial Reconfiguration Decoupler IP core to a project in Vivado® 2020.1 or newer, or when calling `create_ip` to generate a Partial Reconfiguration Decoupler IP core, you will see a message like this:

```
WARNING: [IP_Flow 19-2162] IP 'my_decoupler' is locked:
* IP definition 'Partial Reconfiguration Decoupler (1.0)' for IP
'my_decoupler' has been replaced in the IP Catalog by 'DFX Decoupler
(1.0)'. * IP definition 'Partial Reconfiguration Decoupler (1.0)' for IP
'my_decoupler' (customized with software release 2019.2) has a different
revision in the IP Catalog.
```

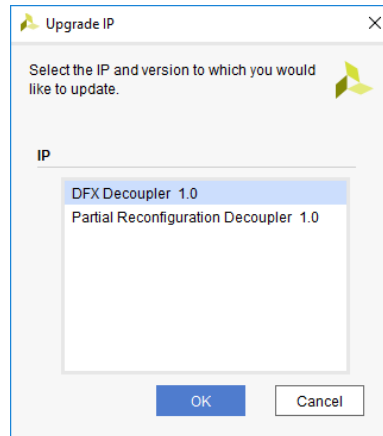
You can perform a direct upgrade from an existing Partial Reconfiguration Decoupler IP instance to the DFX Decoupler core through the standard upgrade process. With a DFX project or a Managed IP project open, select **Reports** → **Report IP Status** to identify any IP in need of upgrading. This IP will appear as locked in its current state.

Figure 20: Locked Status

Source File	IP Status	Recommendation	Change Log	IP Name	Current Version	Recommended Version
my_decoupler	<input type="checkbox"/>	IP definition has been replaced. IP revision change	Upgrade to alternative IP More info	Partial Reconfiguration Decoupler	1.0 (Rev. 8)	DFX Decoupler (1.0)

Check any Partial Reconfiguration Decoupler IP and select **Upgrade Selected**. You will be given a choice of which IP to upgrade to; select the DFX version.

Figure 21: Upgrade IP



The conversion replaces the Partial Reconfiguration Decoupler IP with the equivalent DFX Decoupler IP, with the same set of options and settings. The feature set is identical if upgrading from Partial Reconfiguration Decoupler 1.0 to DFX Decoupler 1.0.

When using the Partial Reconfiguration Decoupler Tcl API capabilities, simply replace any references to `pr_decoupler_v1_0` with `dfx_decoupler_v1_0` in scripts or interactive Tcl use and see the following section for the upgrade code.

Upgrade Code

The following code can be used to make it easier to upgrade the core between versions.

```
if {[dfx_decoupler_v1_0::is_api_compatible dfx_decoupler_v0_0]}
{dfx_decoupler_v1_0::alias_api dfxd
}
```

`is_api_compatible` takes the name of the previous version of the core and returns 1 if the API from the new version is compatible with the API for the old version.

`alias_api <name>` imports all the API commands into a namespace called `<name>`.

Use the following code to migrate from the Partial Reconfiguration Decoupler to the DFX Decoupler.

```
if {[dfx_decoupler_v1_0::is_api_compatible pr_decoupler_v1_0]}
{dfx_decoupler_v1_0::alias_api dfxd
}
```

and follow this with `dfxd::set_property ...` to set the properties for the core.

Debugging

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

If the IP requires a license key, the key must be verified. The Vivado[®] design tools have several license checkpoints for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with an error. License checkpoints are enforced by the following tools:

- Vivado Synthesis
- Vivado Implementation
- write_bitstream (Tcl command)



IMPORTANT! IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.

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. The [Xilinx Community Forums](#) are also available where members can learn, participate, share, and ask questions about Xilinx solutions.

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 [73351](#).

Technical Support

Xilinx provides technical support on the [Xilinx Community Forums](#) 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 ask questions, navigate to the [Xilinx Community Forums](#).

Debug Tools

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

Vivado Design Suite Debug Feature

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

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

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

See the *Vivado Design Suite User Guide: Programming and Debugging* ([UG908](#)).

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 (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open 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 DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the [Design Hubs](#) page.

Note: For more information on DocNav, see the [Documentation Navigator](#) page on the Xilinx website.

References

As well as the resources on the [Dynamic Function eXchange website](#), these documents provide supplemental material useful with this guide:

1. *Dynamic Function eXchange Controller IP LogiCORE IP Product Guide* ([PG374](#))
2. *Dynamic Function eXchange Bitstream Monitor IP LogiCORE IP Product Guide* ([PG376](#))
3. *Dynamic Function eXchange AXI Shutdown Manager IP LogiCORE IP Product Guide* ([PG377](#))
4. *Vivado Design Suite User Guide: Dynamic Function eXchange* ([UG909](#))
5. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* ([UG994](#))
6. *Vivado Design Suite User Guide: Designing with IP* ([UG896](#))
7. *Vivado Design Suite User Guide: Getting Started* ([UG910](#))
8. *Vivado Design Suite User Guide: Logic Simulation* ([UG900](#))
9. *Vivado Design Suite User Guide: Programming and Debugging* ([UG908](#))
10. *Vivado Design Suite User Guide: Implementation* ([UG904](#))
11. *AXI Interconnect LogiCORE IP Product Guide* ([PG059](#))
12. *7 Series FPGAs Clocking Resources User Guide* ([UG472](#))
13. *UltraScale Architecture Clocking Resources User Guide* ([UG572](#))

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
06/03/2020 Version 1.0	
Initial release.	N/A

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

© Copyright 2020 Xilinx, Inc. Xilinx, the Xilinx logo, Alveo, Artix, Kintex, Spartan, Versal, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.