



# **AXI4-Stream Broadcast (Beta Release)**

Version 0.1

May 4, 2023

## Copyright

Copyright © 2021 Rapid Silicon. All rights reserved. This document may not, in whole or part, be reproduced, modified, distributed, or publicly displayed without prior written consent from Rapid Silicon ("Rapid Silicon").

## Trademarks

All Rapid Silicon trademarks are as listed at [www.rapidsilicon.com](http://www.rapidsilicon.com). Synopsys and Synplify Pro are trademarks of Synopsys, Inc. Aldec and Active-HDL are trademarks of Aldec, Inc. Modelsim and Questa are trademarks or registered trademarks of Siemens Industry Software Inc. or its subsidiaries in the United States or other countries. All other trademarks are the property of their respective owners.

## Disclaimers

NO WARRANTIES: THE INFORMATION PROVIDED IN THIS DOCUMENT IS "AS IS" WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING WARRANTIES OF ACCURACY, COMPLETENESS, MERCHANTABILITY, NONINFRINGEMENT OF INTELLECTUAL PROPERTY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL RAPID SILICON OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (WHETHER DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OF OR INABILITY TO USE THE INFORMATION PROVIDED IN THIS DOCUMENT, EVEN IF RAPID SILICON HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF CERTAIN LIABILITY, SOME OF THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

Rapid Silicon may make changes to these materials, specifications, or information, or to the products described herein, at any time without notice. Rapid Silicon makes no commitment to update this documentation. Rapid Silicon reserves the right to discontinue any product or service without notice and assumes no obligation to correct any errors contained herein or to advise any user of this document of any correction if such be made. Rapid Silicon recommends its customers obtain the latest version of the relevant information to establish that the information being relied upon is current and before ordering any products.

# Contents

<b>IP Summary</b>	<b>3</b>
Introduction . . . . .	3
Features . . . . .	3
<b>Overview</b>	<b>4</b>
AXIS Broadcast . . . . .	4
<b>IP Specification</b>	<b>5</b>
Standards . . . . .	5
IP Support Details . . . . .	6
Resource Utilization . . . . .	7
<b>Design Flow</b>	<b>8</b>
IP Customization and Generation . . . . .	8
Parameters Customization . . . . .	9
<b>Example Design</b>	<b>10</b>
Overview . . . . .	10
Simulating the Example Design . . . . .	10
Synthesis and PR . . . . .	10
<b>Test Bench</b>	<b>11</b>
<b>Release</b>	<b>12</b>
Release History . . . . .	12

# IP Summary

## Introduction

A Broadcast is a communication protocol for connecting different components of a system in a parallel and synchronized manner. Broadcast allows multiple slaves to receive the same data from a single master at the same time. This is useful in situations where multiple slaves need to receive the same information, such as when updating the contents of a shared memory. It also improves the system performance and reduces power consumption by avoiding the data duplication and multiple transfers from the master to multiple slaves. This is an AXI-Stream compliant Broadcast IP for easy integration with other AXI based systems.

## Features

- Duplicates one input stream across multiple output streams.
- Support for up to 16 Masters.
- Configurable data width, destination width, ID width and user width.
- Configurable AXI Stream Signals for better control.

# Overview

## AXIS Broadcast

The AXI Broadcast feature is a way for the master to communicate with multiple slaves simultaneously by sending a single transaction to a special broadcast address. This address is usually the highest address in the address space of the system. When the AXI interconnect receives the transaction with the broadcast address, it distributes the transaction to all slaves that are configured to respond to the broadcast address. This feature can be particularly useful in scenarios where multiple slaves need to be updated or accessed simultaneously. For example, in a cache coherency protocol, a processor core may need to invalidate the cache lines of all other cores when it updates a shared memory location. Using the AXI Broadcast feature, the core can send a single transaction with the broadcast address to invalidate all other caches simultaneously. Similarly, in a multicast communication scenario, a master can use the AXI Broadcast feature to send the same data to multiple slaves, such as in a video or audio streaming application. However, it's important to note that using the AXI Broadcast feature can introduce additional complexity and potential contention issues in the system. For example, if multiple masters use the broadcast address to send transactions simultaneously, it can cause contention and delay in the system. Also, not all AXI implementations support the broadcast feature, so it's important to check the specifications of the specific AXI interconnect or IP block being used. A block diagram for the Broadcast IP is shown in Figure 1.

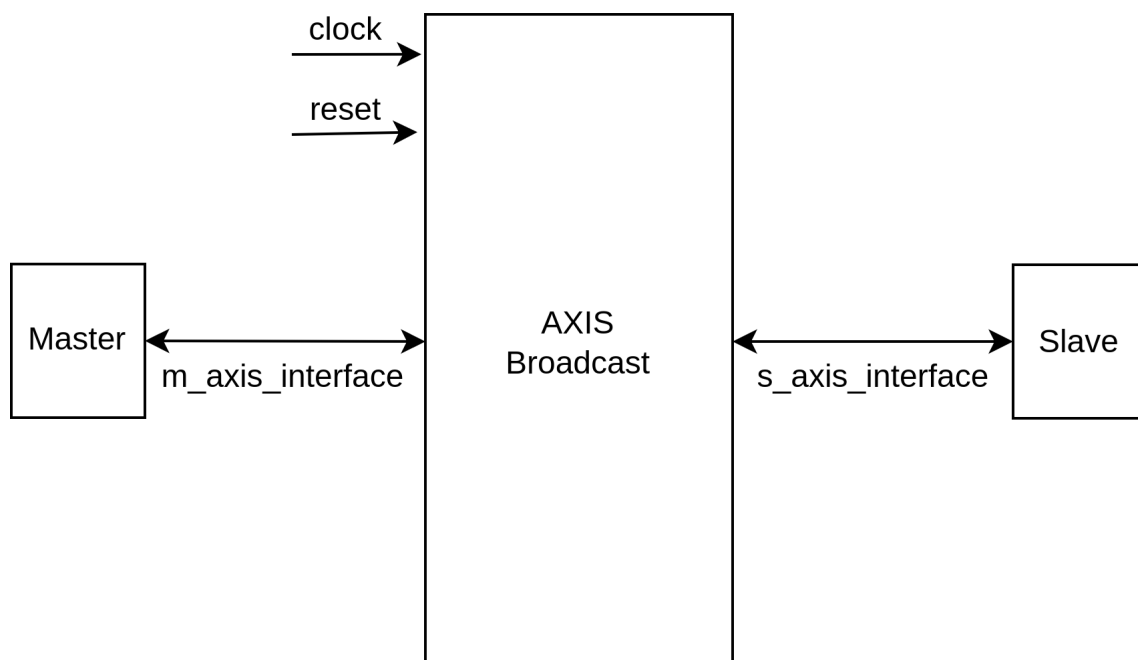


Figure 1: AXIS Broadcast Block Diagram

# IP Specification

In a typical AXI4 Stream transfer, the master device sends a sequence of data packets to the slave device. Each packet includes a data payload and some control information, such as a packet identifier, data length, and end-of-packet marker. The slave device reads the data packets from the master device and processes them according to its own logic.

In AXI4 Stream Broadcast, the master device sends a single stream of data to multiple slave devices simultaneously. This is accomplished by using a broadcast channel, which is a special type of channel that replicates the data stream to multiple output channels.

To implement AXI4 Stream Broadcast, the master device sends the data stream to the broadcast channel, which then replicates the data stream to multiple output channels. Each output channel connects to a separate slave device, which reads the data stream from the channel and processes it according to its own logic.

One of the key benefits of AXI4 Stream Broadcast is that it can help to reduce the amount of data traffic on a system bus, since multiple devices can receive the same data without requiring multiple transfers. This can be especially useful in applications such as video processing, where multiple display devices need to receive the same video stream.

However, there are some challenges associated with implementing AXI4 Stream Broadcast. One challenge is managing the bandwidth and latency of the broadcast channel. Since the channel must replicate the data stream to multiple output channels, it can be more bandwidth-intensive than a regular AXI4 Stream transfer. Additionally, since the output channels may have different processing latencies, it is important to ensure that all slave devices receive the data stream correctly and in a timely manner. This Broadcast IP supports up to 16 masters with much customization suited for consumer needs in an FPGA friendly fashion.

Overall, AXI4 Stream Broadcast is a useful protocol for streaming data to multiple devices in a digital system. It can help to reduce system complexity and data traffic, while enabling efficient and reliable data transfers. The internal block diagram can be seen in Figure 2.

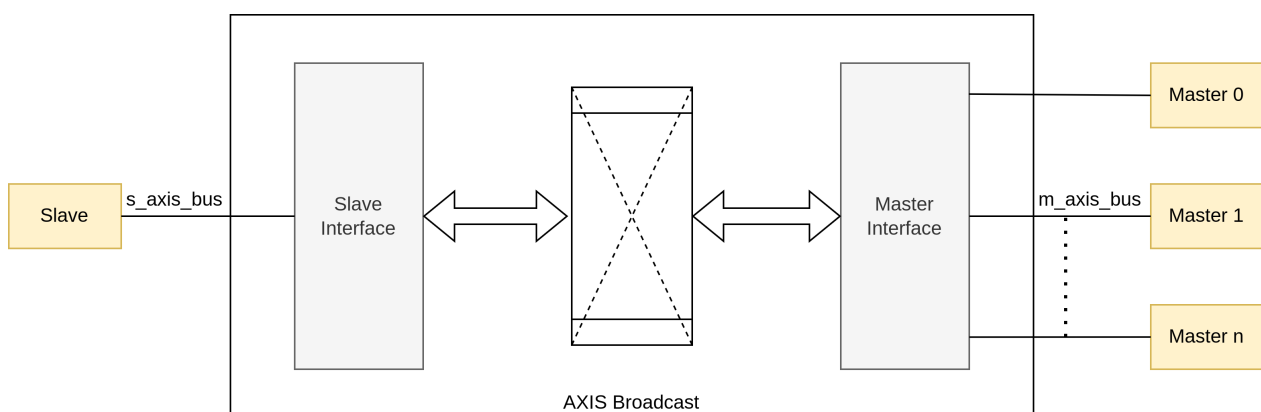


Figure 2: AXIS Broadcast Internal Diagram

## Standards

The AXI4-Stream interface is compliant with the AMBA® AXI Protocol Specification.

## IP Support Details

The Table 1 gives the support details for AXIS Broadcast.

Compliance		IP Resources				Tool Flow		
Device	Interface	Source Files	Constraint Files	Testbench	Simulation Model	Analyze and Elaboration	Simulation	Synthesis
Gemini	AXI4-Stream	Verilog	-	Python	Cocotb	Verific (Raptor)	Icarus (Raptor)	Raptor

Table 1: IP Details

## Port List

Table 2 lists the top interface ports of the AXIS Broadcast.

Signal Name	I/O	Description
<b>AXI Clock and Reset</b>		
clk	I	AXI4-Stream Clock
rst	I	AXI4-Stream RESET
<b>AXI Slave Interface</b>		
s_axis_tdata	I	AXI4-Stream data
s_axis_tkeep	I	AXI4-Stream keep data qualifier
s_axis_tvalid	I	AXI4-Stream valid transfer
s_axis_tready	O	AXI4-Stream transfer ready
s_axis_tlast	I	AXI4-Stream boundary of transfer packet
s_axis_tid	I	AXI4-Stream data stream identifier
s_axis_tdest	I	AXI4-Stream data routing information
s_axis_tuser	I	AXI4-Stream user defined sideband information
<b>AXI Master Interface</b>		
m{}_axis_tdata	O	AXI4-Stream data
m{}_axis_tkeep	O	AXI4-Stream keep data qualifier
m{}_axis_tvalid	O	AXI4-Stream valid transfer
m{}_axis_tready	I	AXI4-Stream transfer ready
m{}_axis_tlast	O	AXI4-Stream boundary of transfer packet
m{}_axis_tid	O	AXI4-Stream data stream identifier
m{}_axis_tdest	O	AXI4-Stream data routing information
m{}_axis_tuser	O	AXI4-Stream user defined sideband information

Table 2: AXIS Broadcast Interface

## Parameters

Table 3 lists the parameters of the AXIS Broadcast.

Parameter	Values	Default Value	Description
M COUNT	2 - 16	4	Number of Master Interfaces
DATA WIDTH	1 - 4096	8	Data Width for each transfer
USER WIDTH	1 - 4096	1	Data Width for user defined sideband information
DEST WIDTH	1 - 32	1	Destination Width
ID WIDTH	1 - 32	2	ID Width
ID ENABLE	0 / 1	0	ID Enable
USER ENABLE	0 / 1	1	User Data Enable
DEST ENABLE	0 / 1	1	TDEST Signal Enable
LAST ENABLE	0 / 1	1	TLAST Signal Enable

Table 3: Parameters

## Resource Utilization

The parameters for computing the maximum and minimum resource utilization are given in Table 4, remaining parameters have been kept at their default values.

Tool	Raptor Design Suite			
FPGA Device	GEMINI			
Configuration			Resource Utilization	
Minumum Resource	Options	Configuration	Resource	Utilized
	M COUNT	2	LUTs	10
	DATA WIDTH	1		
	ID ENABLE	0		
	DEST ENABLE	0	Registers	8
	USER ENABLE	0		
	LAST ENABLE	0		
Maximum Resource	Options	Configuration	Resource	Utilized
	M COUNT	16	LUTs	8801
	DATA Width	4096		
	ID WIDHT	32		
	DEST WIDTH	32	Registers	17556
	USER WIDTH	4096		
	LAST ENABLE	1		

Table 4: Resource Utilization



# Design Flow

## IP Customization and Generation

AXIS Broadcast IP core is a part of the Raptor Design Suite Software. A customized AXIS Broadcast can be generated from the Raptor's IP configurator window as shown in Figure 3.

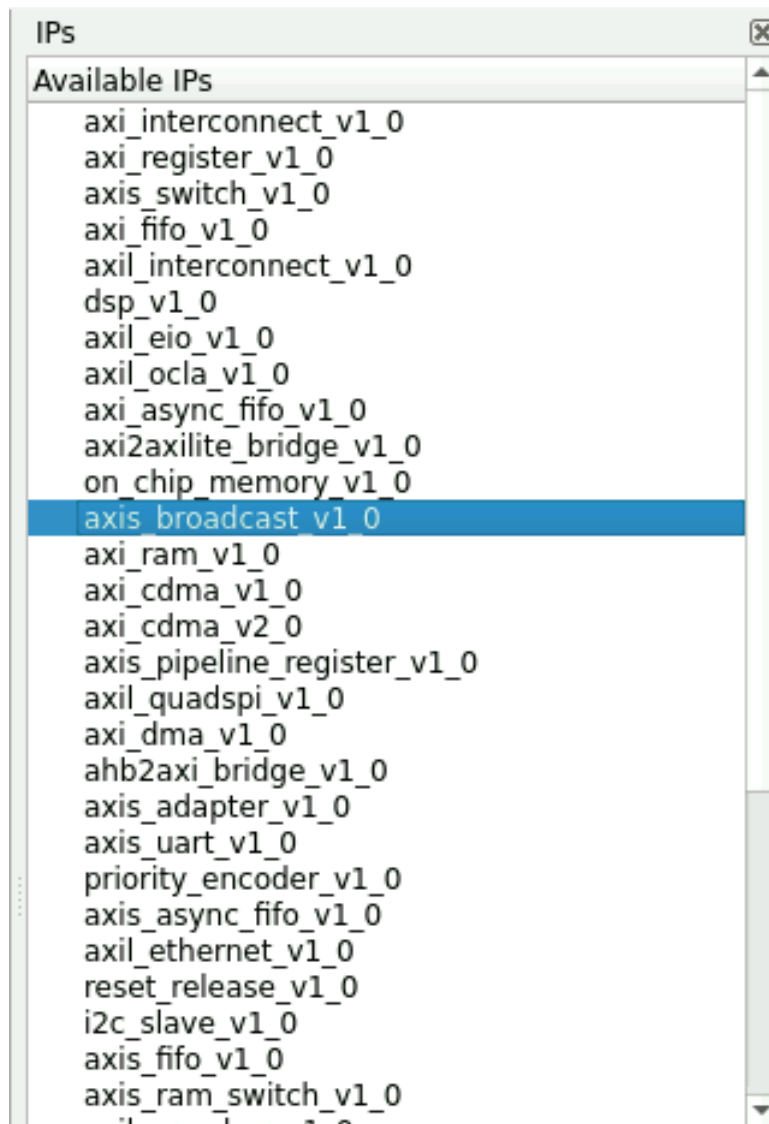
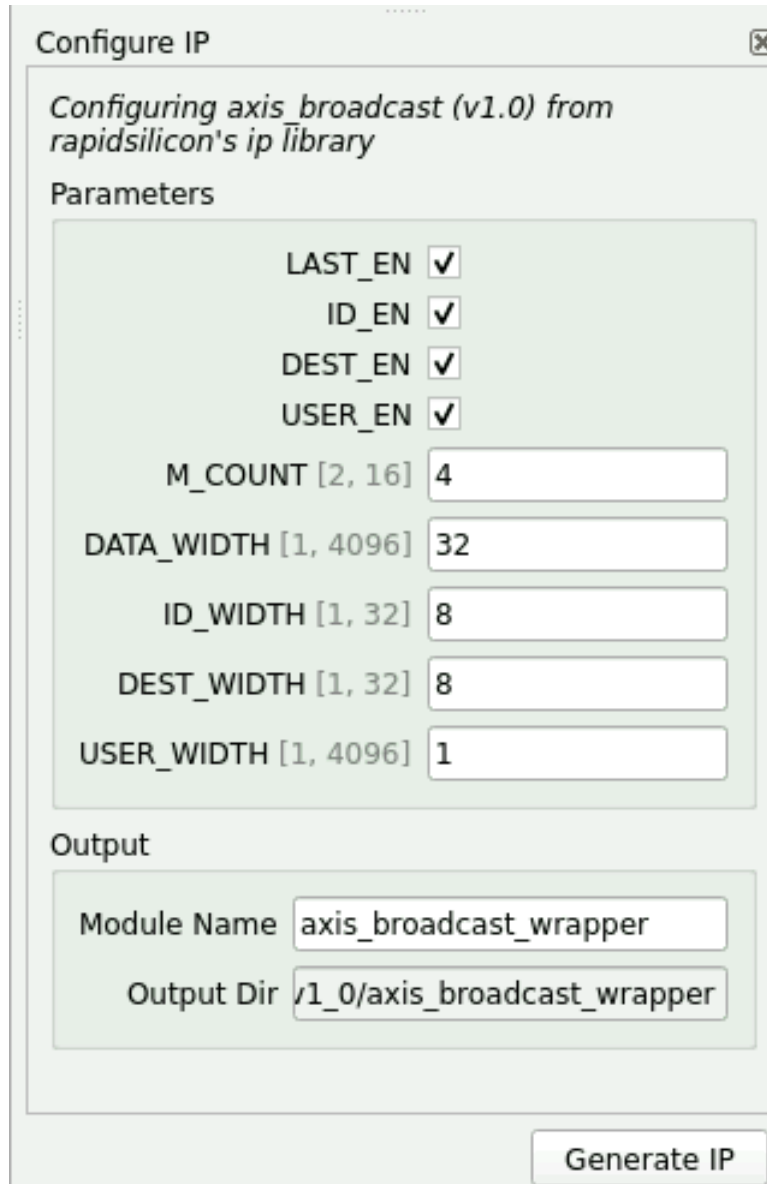


Figure 3: IP list

## Parameters Customization

From the IP configuration window, the parameters of the Broadcast can be configured and Broadcast features can be enabled for generating a customized Broadcast IP core that suits the user application requirement as shown in Figure 4. After IP Customization, all the source files are made available to the user with a top wrapper that instantiates a parameterized instance of the AXIS Broadcast.



**Configure IP**

*Configuring axis\_broadcast (v1.0) from rapidsilicon's ip library*

**Parameters**

LAST\_EN ☒

ID\_EN ☒

DEST\_EN ☒

USER\_EN ☒

M\_COUNT [2, 16]

DATA\_WIDTH [1, 4096]

ID\_WIDTH [1, 32]

DEST\_WIDTH [1, 32]

USER\_WIDTH [1, 4096]

**Output**

Module Name

Output Dir

**Generate IP**

Figure 4: IP Configuration

# Example Design

## Overview

This AXIS Broadcast IP can be utilized in any system that has multiple masters and there is a need to forward the same data to all of these masters from one slave. This helps reduce redundancy of similar data by providing all masters with the same data traffic. One such example design of this AXIS Broadcast can be visualized in Figure 5.

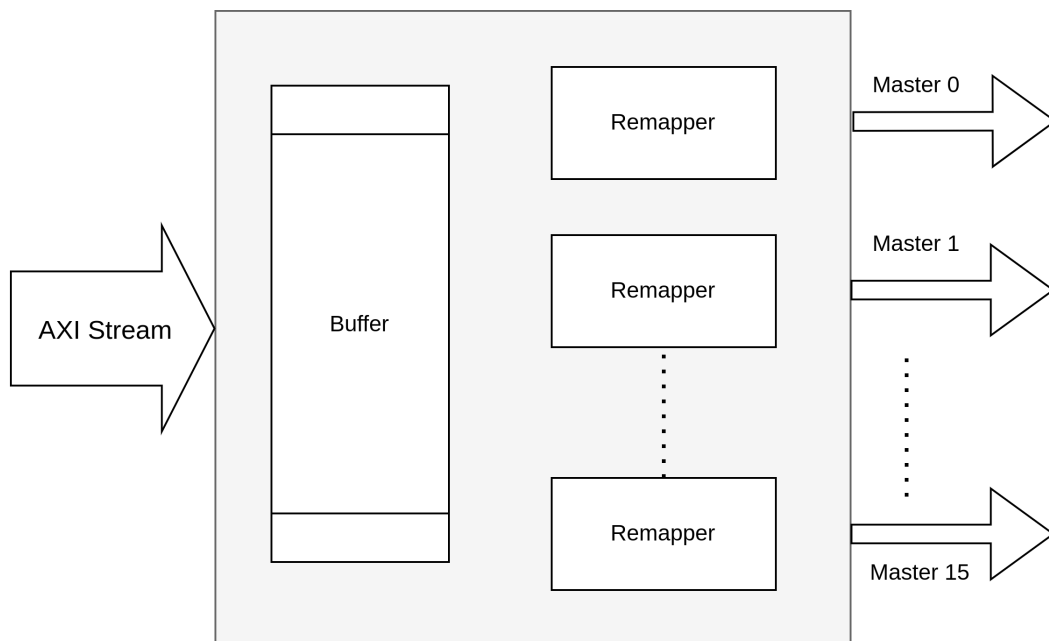


Figure 5: AXIS Broadcast replicating one into multiple Streams

## Simulating the Example Design

The IP being Verilog HDL, can be simulated via a bunch of industry standard stimulus. For instance, it could be simulated via writing a Verilog Test-bench, or incorporating a soft processor that can stimulate this Broadcast IP. The bundled example design is stimulated via a Coco-tb based environment that iteratively stimulates all the master/slave pairs while also stress testing the data routing between them.

## Synthesis and PR

Raptor Suite is armed with tools for Synthesis along with Post and Route capabilities and the generated postsynthesis and post-route and place netlists can be viewed and analyzed from within the Raptor. The generated bitstream can then be uploaded on an FPGA device to be utilized in hardware applications.

## Test Bench

A Coco-tb based test bench can be found in the **/sim** repository formed after the generation of the IP. It generates a Broadcast IP with the required number of Masters. This test environment can be simulated with any Verilog HDL simulator of choice e.g., Verilator or Icarus. This simulates the generated IP under various test conditions including stimulating all slave interfaces individually and then some stress test cases where all the interfaces are stimulated together. This makes up for a total of 4 tests upon passing of which the IP can be verified functionally.

# Release

## Release History

Date	Version	Revisions
May 4, 2023	0.1	Initial version AXI Stream Broadcast User Guide Document