AXI4 DMA
(Beta Release)

Version 1.0

December 5, 2023

Copyright

Trademarks

Disclaimers

Contents

IP Summary

Introduction

DMA (Direct Memory Access) is a feature of digital systems that allows data to be transferred directly between memory and peripheral devices, without involving the CPU. This can greatly improve performance and reduce the load on the CPU, as the CPU does not have to be involved in the data transfer process. DMA is commonly used for tasks such as transferring audio and video data, and for data transfer between devices such as storage drives and network interfaces. There are different types of DMA controllers, such as the AXI DMA, which uses the AXI bus for communication that allows for easy integration with other AXI based systems. This AXI DMA utilizes the AXI Protocol for the configuration of descriptors. This is an AXI 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.

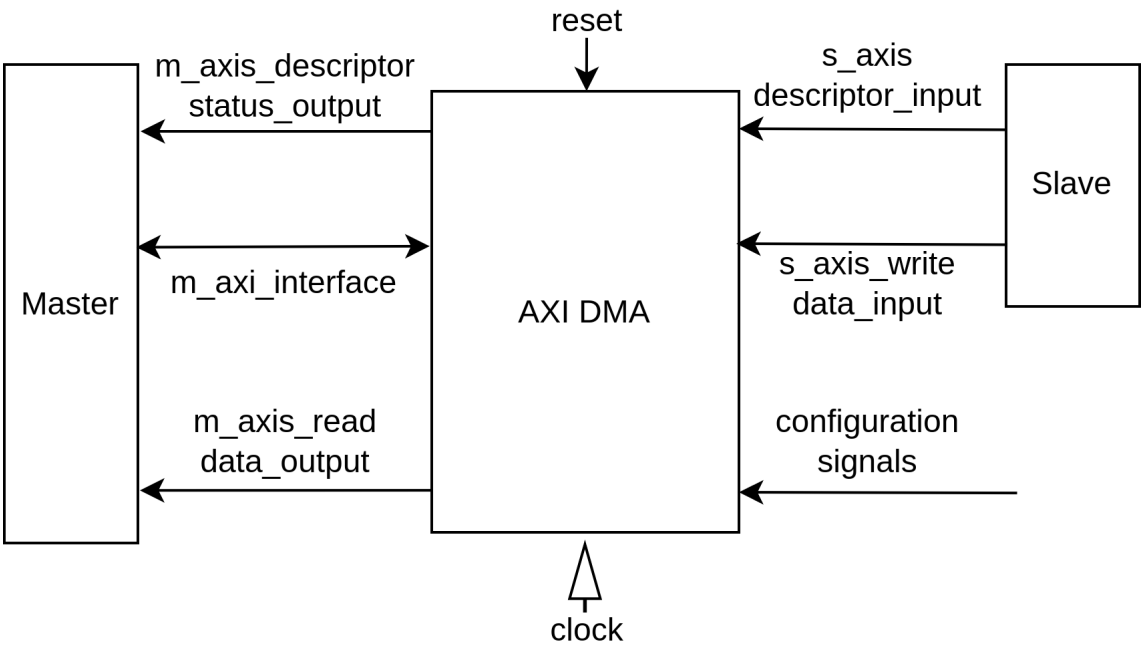

Figure 1: AXIS Broadcast Block Diagram

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.

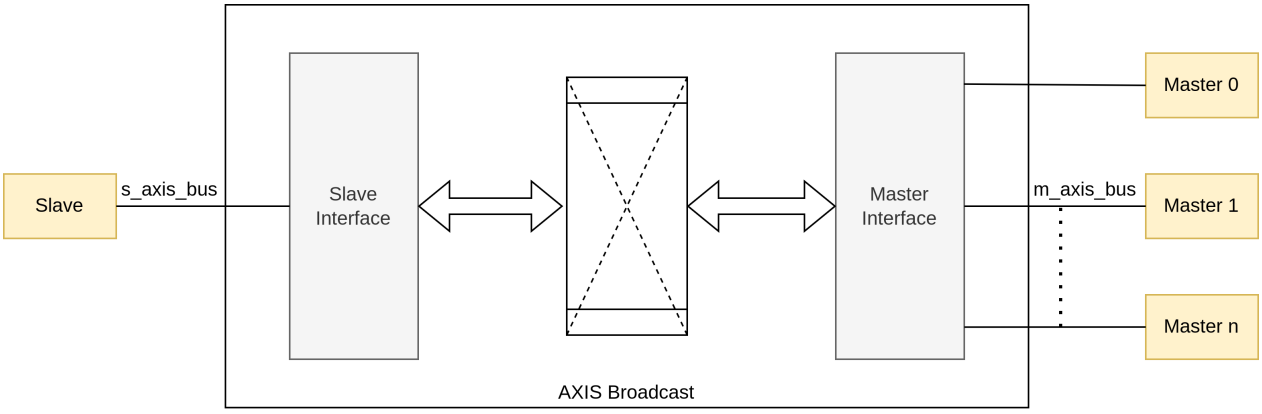

Figure 2: AXIS Broadcast Internal Diagram

Standards

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

IP Support Details

The Table 1 gives the support details for AXI DMA.

Compliance		IP Resources					Tool Flow		
Device	Interface	Source Files	Constraint File	Testbench	Simulation Model	Software Driver	Analyze and Elaboration	Simulation	Synthesis
GEMINI	AXI4	Verilog	SDC	Python	Cocotb	Icarus	Raptor	Raptor	Raptor

Table 1: IP Details

Port List

Table 2 lists the top interface ports of the AXI DMA.

Signal Name	I/O	Description
AXI Clock and Reset		
clk	I	AXI4-Stream Clock
rst	I	AXI4-Stream RESET
AXI Slave Interface		
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
AXI Master Interface		
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 AXI DMA.

Parameter	Values	Default Value	Description
AXI DATA WIDTH	8, 16, 32, 64, 128, 256	32	DMA Data Width
AXI ADDR WIDTH	8 - 16	8	DMA Address Width
AXI ID WIDTH	1 - 32	8	DMA ID Width
AXI MAX BURST LEN	1 - 256	16	DMA AXI burst length
AXIS LAST ENABLE	0 / 1	1	AXI stream tlast
AXIS ID ENABLE	0 / 1	0	AXI stream tid
AXIS ID WIDTH	1 - 32	8	DMA AXI stream tid width
AXIS DEST ENABLE	0 / 1	0	AXI stream tdest
AXIS DEST WIDTH	1 - 8	8	DMA AXI stream tdest width
AXIS USER ENABLE	0 / 1	1	AXI stream tuser
AXIS USER WIDTH	1 - 8	1	DMA AXIS User Width
LEN WIDTH	1 - 20	20	DMA AXI Width of length field
TAG WIDTH	1 - 8	8	DMA Width of tag field
ENABLE SG	0 / 1	0	Support for scatter/gather DMA
ENABLE UNALIGNED	0 / 1	0	Support for unaligned transfers
IP TYPE	-	AXI_DMA	Type of Peripheral
IP VERSION	-	<ip_version>	Version of Peripheral
IP ID	-	<date_and_time>	Date and Time of the generated Peripheral

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 Utilized**	
Minimum Resource	**Options**	**Configuration**	**Resources**	**Utilized**
	AXI DATA WIDTH	8	LUTs BRAM REGISTERS	328 2 328
	AXI ADDR WIDTH	8		
	AXI ID WIDTH	1		
	AXI MAX BURST LEN	1		
	AXIS LAST ENABLE	0		
	AXIS ID ENABLE	0		
	AXIS DEST ENABLE	0		
	AXIS USER ENABLE	0		
	LEN WIDTH	1		
	TAG WIDTH	1		
	ENABLE SG	0		
	ENABLE UNALIGNED	0		
Maximum Resource	**Options**	**Configuration**	**Resources**	**Utilized**
	AXI DATA WIDTH	256	LUTs BRAM REGISTERS ADDER CARRY	4555 24 1217 140
	AXI ADDR WIDTH	16		
	AXI ID WIDTH	32		
	AXI MAX BURST LEN	256		
	AXIS LAST ENABLE	1		
	AXIS ID ENABLE	1		
	AXIS ID WIDTH	32		
	AXIS DEST ENABLE	1		
	AXIS DEST WIDTH	8		
	AXIS USER ENABLE	1		
	AXIS USER WIDTH	8		
	LEN WIDTH	20		
	TAG WIDTH	8		
	ENABLE SG	1		
	ENABLE UNALIGNED	1		

Table 4: Resource Utilization

Design Flow

IP Customization and Generation

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

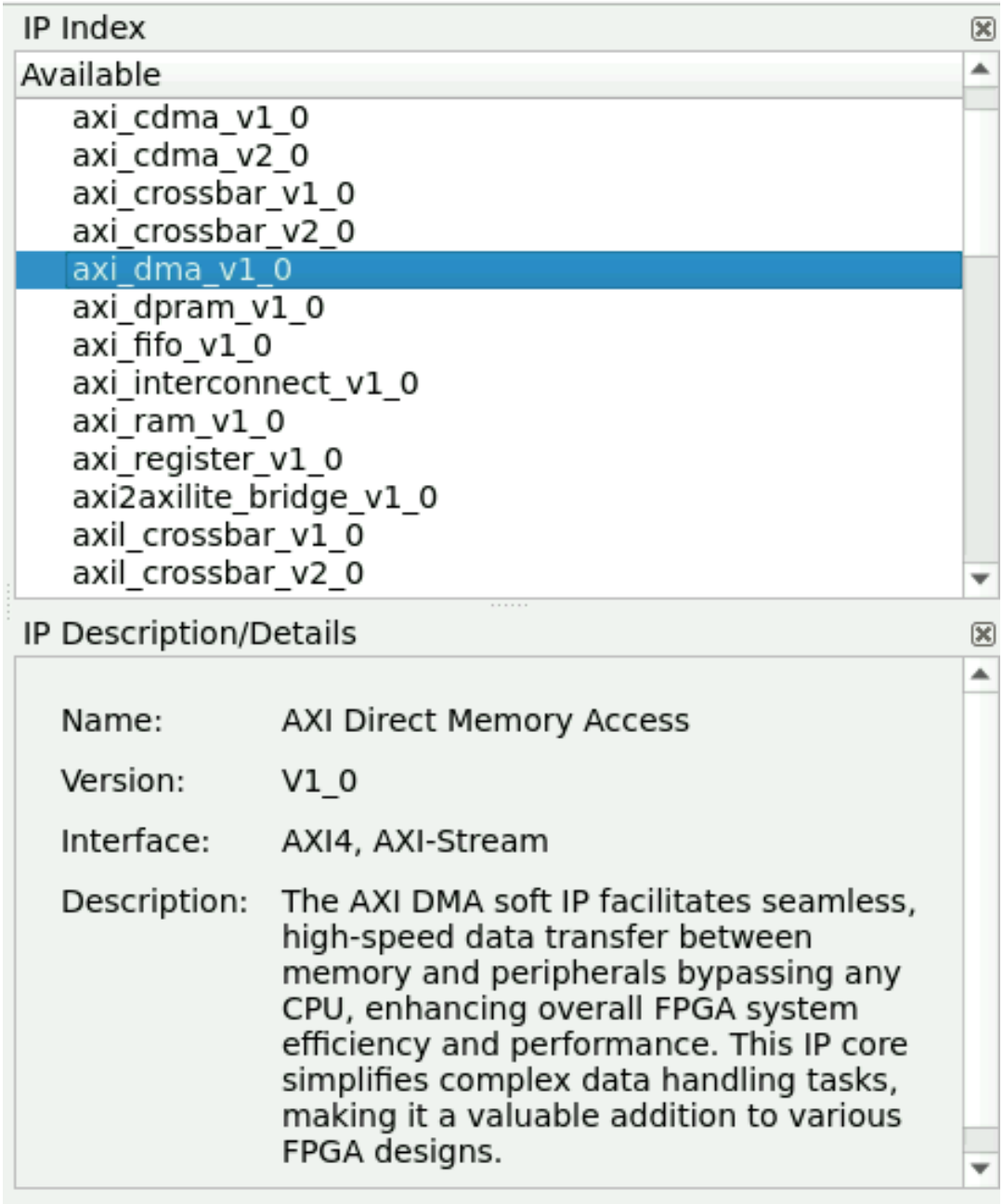

Figure 3: IP list

Parameters Customization

From the IP configuration window, the parameters of the DMA can be configured and DMA features can be enabled for generating a customized DMA 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 AXI DMA.

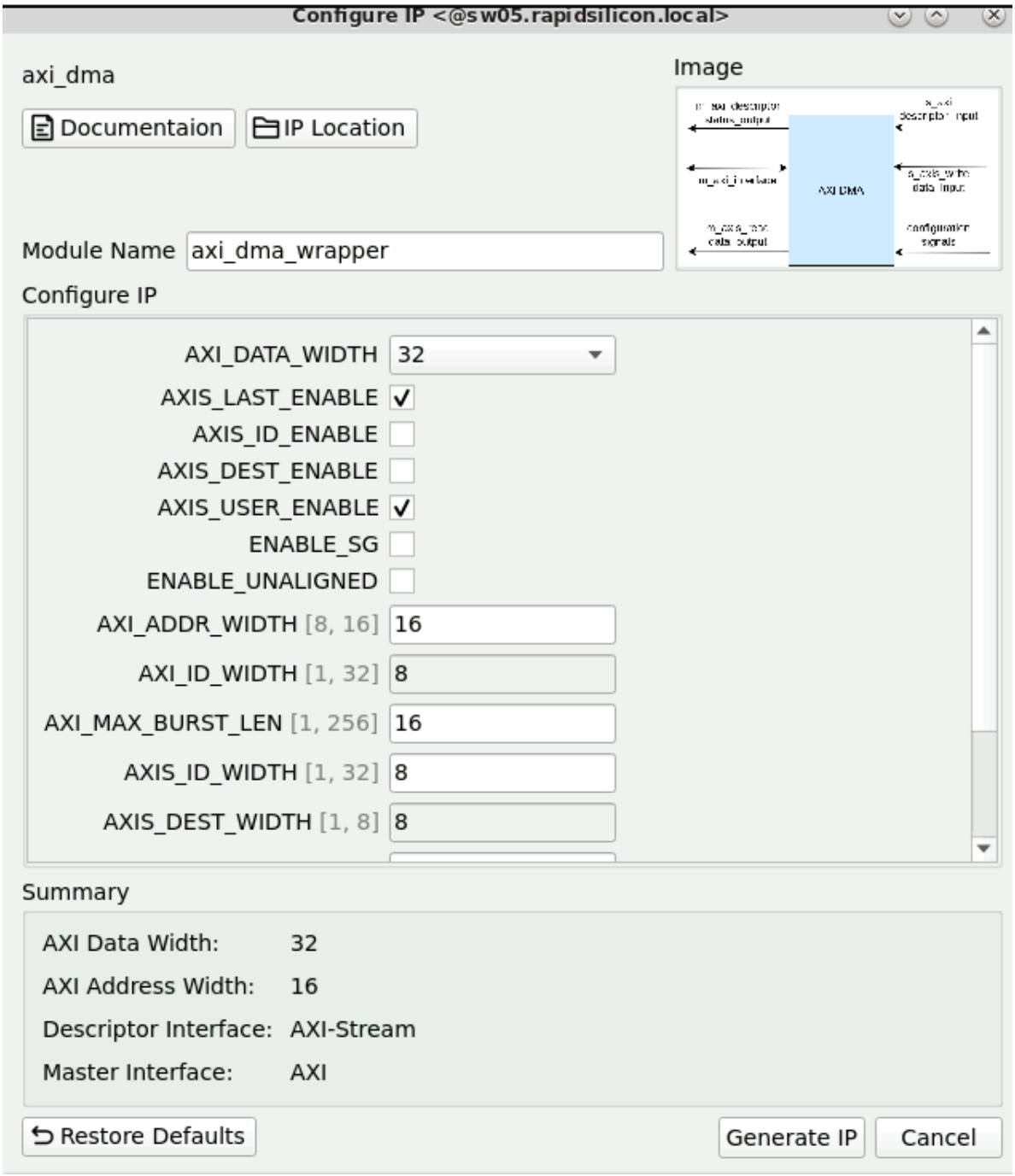

Figure 4: IP Configuration

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.

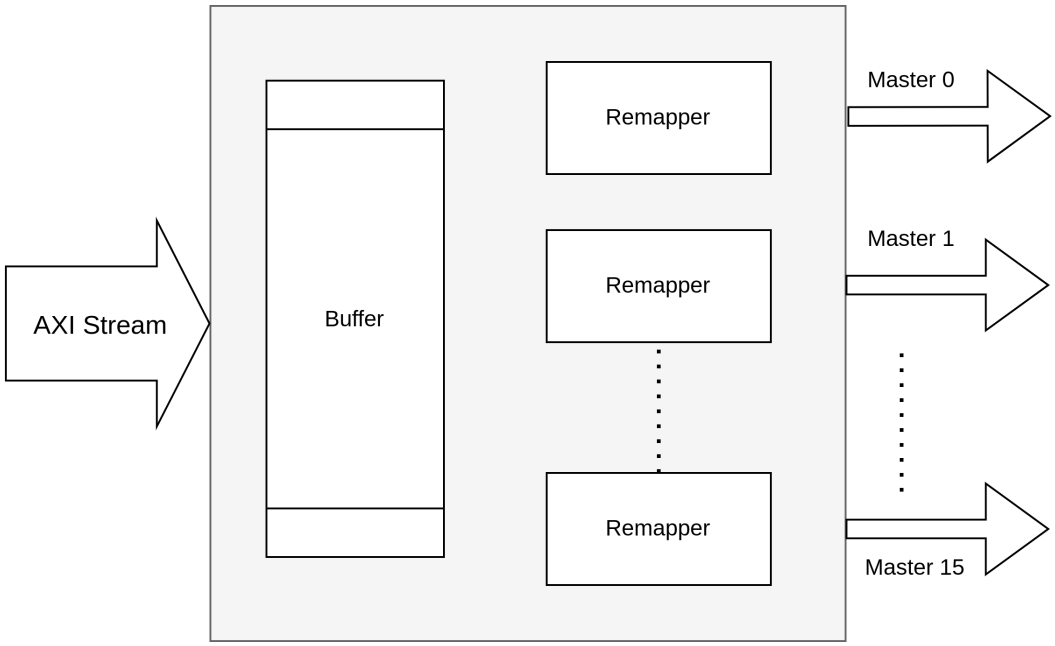

Figure 5: AXIS Broadcast replicating one into multiple Streams

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 DMA. The bundled example design is stimulated via a Coco-tb based environment that iteratively stimulates individual reads and write operations via the DMA to completely check its functionality.

Synthesis and PR

Raptor Suite is armed with tools for Synthesis along with Post and Route capabilities and the generated post synthesis and post-route and place net-lists can be viewed and analyzed from within the Raptor. The generated bit-stream 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 parameters suited for the user application requirement. 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 individual read and write tests. This makes up for a total of 8 tests upon passing of which the IP can be verified functionally. Being a python based IP and being able to be simulated on a number of bundled simulators, this makes for a versatile testing experience making sure most of the IP gets covered. The simulation can be easily run by clicking the "Simulate IP" button as shown in figure 6.

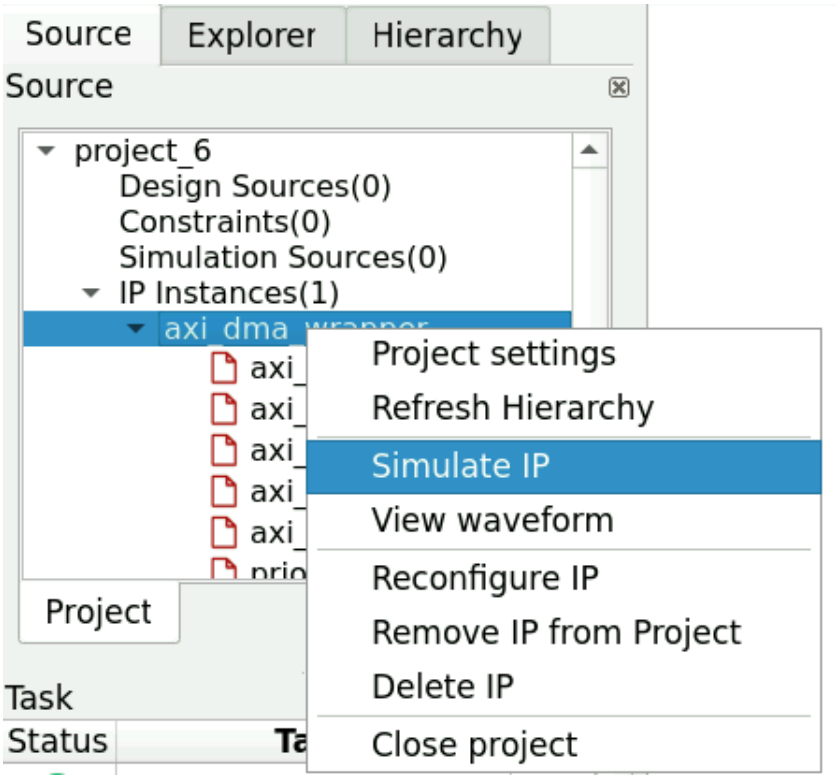

Figure 6: Simulate IP Window

The waveforms are also dumped in-depth analysis of the whole operation which can be seen by clicking the "View Waveform" button as shown in 7.

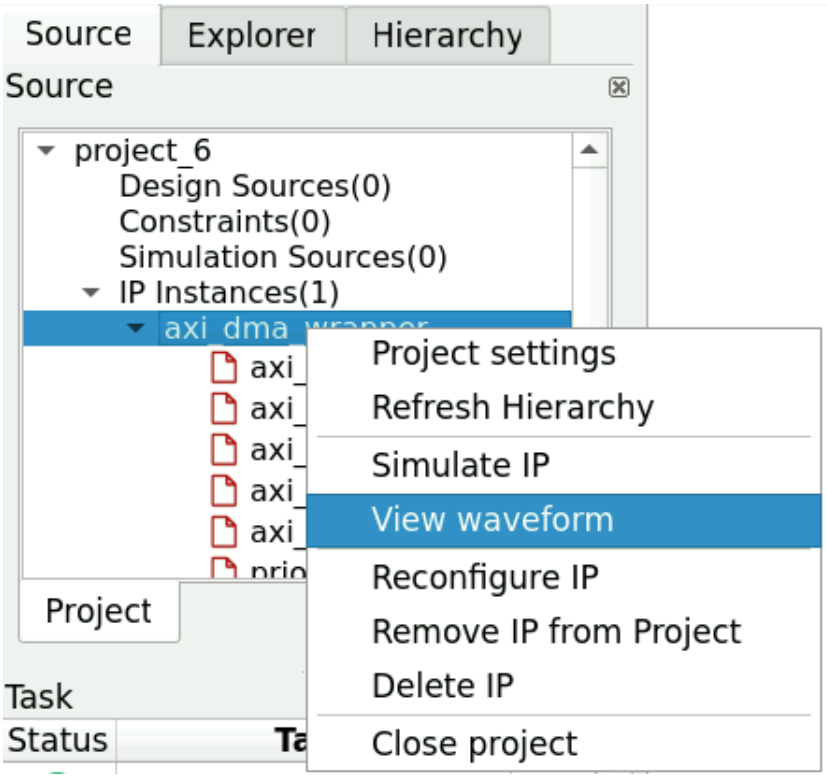

Figure 7: View Waveform Window

The simulation results are also displayed in the console window a glimpse of which can be seen in figure 8.

```
**********************************************************************************************
** TEST                                STATUS  SIM TIME (ns)  REAL TIME (s)  RATIO (ns/s) **
**********************************************************************************************
** test_axi_dma.run_test_write_001    PASS       55530.00           2.54       21894.29  **
** test_axi_dma.run_test_write_002    PASS      111840.00           4.70       23781.37  **
** test_axi_dma.run_test_write_003    PASS       99940.00           4.29       23306.88  **
** test_axi_dma.run_test_write_004    PASS      111860.00           4.98       22459.56  **
** test_axi_dma.run_test_read_001     PASS        7850.00           0.40       19786.87  **
** test_axi_dma.run_test_read_002     PASS       14330.00           0.63       22834.07  **
** test_axi_dma.run_test_read_003     PASS       14350.00           0.63       22788.33  **
** test_axi_dma.run_test_read_004     PASS       19730.00           0.88       22455.58  **
**********************************************************************************************
** TESTS=8 PASS=8 FAIL=0 SKIP=0                435430.01          20.93       20799.47  **
**********************************************************************************************
```

Figure 8: Simulation Results

Release

Release History

Date	Version	Revisions
December 5, 2023	0.1	Initial version AXI DMA User Guide Document