Datasheet

Audio Interfaces Total IP Solution

SLIMbus
SoundWire
I2S

Arasan Chip Systems Inc.
2010 North First Street, Suite #510, San Jose, CA 95131
Ph: 408-282-1600
Fax: 408-282-7800
www.arasan.com

Disclaimer

This document is written in good faith with the intent to assist the readers in the use of the product. Circuit diagrams and other information relating to Arasan Chip Systems' products are included as a means of illustrating typical applications. Although the information has been checked and is believed to be accurate, no responsibility is assumed for inaccuracies. Information contained in this document is subject to continuous improvement and development.

Arasan Chip Systems' products are not designed, intended, authorized or warranted for use in any life support or other application where product failure could cause or contribute to personal injury or severe property damage. Any and all such uses without prior written approval of an Officer of Arasan Chip Systems Inc. will be fully at the risk of the customer.

Arasan Chip Systems Inc. disclaims and excludes any and all warranties, including, without limitation, any and all implied warranties of merchantability, fitness for a particular purpose, title, and infringement and the like, and any and all warranties arising from any course or dealing or usage of trade.

This document may not be copied, reproduced, or transmitted to others in any manner. Nor may any use of information in this document be made, except for the specific purposes for which it is transmitted to the recipient, without the prior written consent of Arasan Chip Systems Inc. This specification is subject to change at any time without notice. Arasan Chip Systems Inc. is not responsible for any errors contained herein.

In no event shall Arasan Chip Systems Inc. be liable for any direct, indirect, incidental, special, punitive, or consequential damages; or for loss of data, profits, savings or revenues of any kind; regardless of the form of action, whether based on contract; tort; negligence of Arasan Chip Systems Inc or others; strict liability; breach of warranty; or otherwise; whether or not any remedy of buyers is held to have failed of its essential purpose, and whether or not Arasan Chip Systems Inc. has been advised of the possibility of such damages.

Restricted Rights

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Copyright Notice

No part of this specification may be reproduced in any form or means, without the prior written consent of Arasan Chip Systems, Inc.

Questions or comments may be directed to:

Arasan Chip Systems Inc.
2010 North First Street, Suite 510
San Jose, CA 95131
Ph: 408-282-1600
Fax: 408-282-7800
Email: sales@arasan.com

Contents

Copyright © 2015, Arasan Chip Systems Inc.

Figures

Copyright © 2015, Arasan Chip Systems Inc.

1 Introduction

MIPI Alliance provides the Serial Low-power Inter-chip Media Bus (SLIMbus) and SoundWire, a pair of optimized interfaces with both complementary and unique features to integrate audio devices in a mobile or mobile-inspired system. SLIMbus and SoundWire are designed to replace older legacy interface designs that present limitations to system designers, whether in terms of power, pin count, ease of integration and consistency of design from one system to another, or lack of scalability. Both can coexist in a system or with non-MIPI interfaces through bridging solutions.

The MIPI alliance has defined the SLIMbus specification which provides a standard, robust, scalable, low-power, high-speed, cost-effective, two wire multipurpose interface that supports a wide range of digital audio and control solutions for mobile terminals. Soundwire is suited for small, cost-sensitive audio peripherals such as modern digital class-D amplifiers anddigital microphones.

The SoundWireSpecification describes an interface for transporting audio and related data. The SoundWire interface addresses a wide range of system scenarios where up to eleven Slave interfaces share the common bus, and where those Slaves contain anything from one single-channel audio Data Port up to fourteen 8-channel Data Ports (using multi-lane extension up to 8 lanes to provide sufficient bandwidth).

1.1 Arasan’s Contribution to MIPI

Arasan has been a member of MIPI for over ten years. We are active participants in a number of working groups. We work closely with other member customers to ensure compliant implementation of standards based IP.

1.2 Arasan’s Audio Interfaces Offerings

Arasan brings expertise in serial audio and over one hundred design wins with MIPI® SLIMbusSM to the MIPI SoundWireSM standard.

The Arasan Chip Systems MIPI SLIMbus digital controller provides a mature, highly capable solution for the MIPI SLIMbus Protocol. Arasan provides customers with both Host and Device functionality used on multiple production designs, the MIPI SLIMbus controller IP is applicable for host and application processors.

The Total MIPI SoundWire IP Solution from Arasan enables early adopters the fastestpath to adoption of this new standard by offering a comprehensive IP package that includes the Verilog RTLsource code for Master and Slave, fully validated for compliance with the standard, a comprehensive testenvironment with a compliance suite for verification of the IP package, a SoundWire Hardware DevelopmentKit (“HDK”) for FPGA prototyping, a SoundWire protocol analyzer and a complete SoundWire software stack.

Arasan also has a diverse portfolio of connectivity IP products including SPI, I2C, I2S and UART. These protocols are vital for the integration of SoCs with peripheral chipsets in order to form a complete hardware platform. They are frequently used by the SoC to configure, control and gather diagnostic information at the platform level to ensure correct operation of the hardware. The Arasan I2S Controller IP Core is a two-channel I2S serial audio controller compliant to the Philips* Inter-IC Sound specification. The I2S bus is used for connecting audio components such as speakers, DACs, or audio subsystems.

1.3 Arasan's Total IP Solution

Arasan provides a Total IP Solution, which encompasses all aspects of IP development and integration, including analog and digital IP cores, verification IP, software stacks & drivers, and hardware validation platforms.Benefits of Total IP Solution:

- Seamless integration from PHY to Software
- Assured compliance across all components
- Single point of support
- Easiest acquisition process (one licensing source)
- Lowest overall cost including cost of integration
- Lowest risk for fast time to market

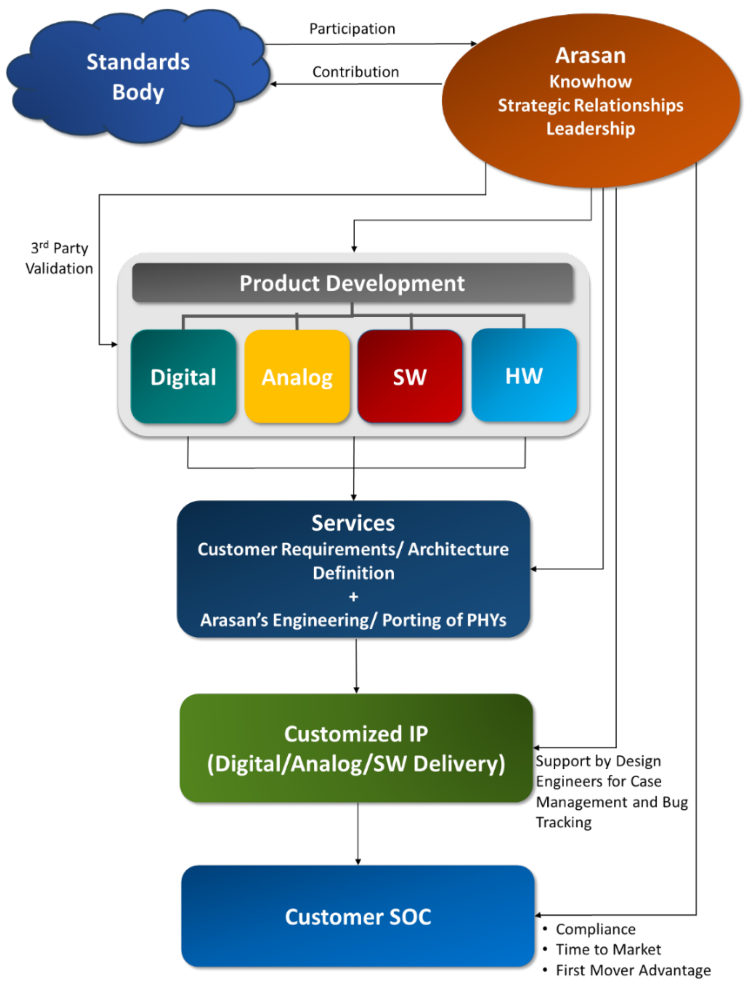

Figure 1: Arasan's Total IP Solution

2 SLIMbus Host Controller IP

2.1 Overview

Arasan Chip Systems is a leading SoC IP provider of a complete suite of MIPI compliant IP solutions which consist of analog PHY and digital controller IP Cores, Verification IP, Software stacks and drivers, Hardware Validation Platforms for software development and Compliance Testing, and optional customization services.

The MIPI compliant IP Cores serve as interface building blocks that simplify sub-system level interconnect architectures in mobile platforms. This leads to smaller footprint, greater interoperability between mobile IP, chips and devices from diverse sources, and lower power and EMI.

This document describes the Arasan IP Core that functions as a MIPI SLIMbus Host Controller, which typically resides in a mobile platform's application processor and provides two-wire, multi-drop connectivity with multiple audio and other Low/Mid bandwidth peripheral devices.

2.2 Features

- Compliant with MIPI SLIMbus specification version 1.1
  - Delivered in Reuse Methodology Manual (RMM) compliant Verilog RTL format
  - Small footprint

- Contains full-featured active interface device, with support for
  - Dynamic channel allocation and management
  - All SLIMbus Core Messages
  - Error handling and recovery, as defined in the MIPI SLIMbus specification

- Embedded Framer which can be active or passive, and supports SLIMbus clock generation handoff To/From Framers in other components, with support for
  - Clock Gears 1 to 10, either as clock source or clock receiver
  - Maximum of 28.8 MHz bit-serial rate
  - Dynamic SLIMbus clock frequency scaling and clock Pause/Resume minimizes power

- Contains one generic device with up to 8 active port pairs; each port supports
  - Isochronous, Pushed, Pulled, Asynchronous Simplex, Extended Asynchronous Simplex and Locked transport protocols over SLIMbus with data segment size of 1 to 31 slots
  - Interface options to application processor
  - Generic FIFO Interface
  - Audio or peripheral data sample sizes of 8,16, 24 or 32 bits
  - Different port pairs can have different sampling rates

- Slave DMA Interface
  - Each generic port has separate channel to external DMA master
  - Interrupt wake-up mechanism to turn system bus interface on when SLIMbus host ready to Source/Sink system memory data

- 32 bit AHB 2.0 Slave Interface to CPU/Memory sub-system
  - System Clock frequency range from 66 to 150 MHz
  - Used by driver to configure the IP using Programmed IO
  - Optionally used for data transfer To/From generic device

2.3 Architecture

2.3.1 Functional Description

The Arasan SLIMbus Host Controller IP is designed to provide MIPI SLIMbus 1.1 compliant connectivity to a SoC.

As a SLIMbus host, this IP is responsible for the establishment, maintenance and shutdown of the entire SLIMbus system under control of the host software Drivers/Stack and in response to the presence and bandwidth requirements of the various SLIMbus devices on the bus.

SLIMbus has a TDM channel allocation structure for control messages and data. When its Framer is active, the host drives the SLIMbus clock, creates the SLIMbus frames, and enables the various devices to synchronize and share the available bandwidth.

This IP contains a configurable generic device that transfers data to and from remote SLIMbus components and the SoC's system memory. Alternately, one or more port pairs of the generic device can be used by SLIMbus host component to incorporate bridge functions to legacy interfaces, like I2S, I2C and SPI, or interface directly to audio DAC's and ADC's.

2.3.2 Functional Block Diagram

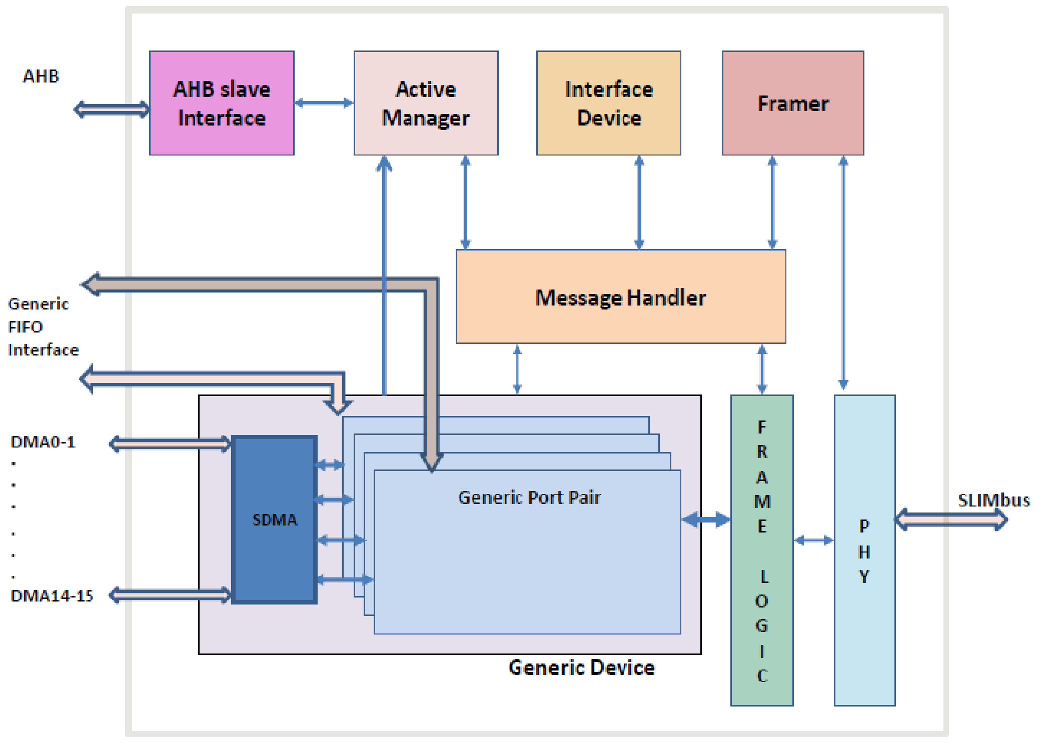

Figure 2: SLIMbus Host Functional Block Diagram

2.3.3 Functional Block Diagram Description

2.3.3.1 Slave Interface

The AHB Slave Interface block is the programming interface for the software drivers to receive and send control messages for the SLIMbus system and configures the IP through a combination of interrupts and programmed IO. In addition, this block performs data transfers To/From system memory and the IP's internal Generic Device when running in DMA mode of operation.

2.3.3.2 Active Manager

Active Manager is responsible for the administration of the SLIMbus namely, enumeration of the SLIMbus devices, bus configuration, and bandwidth allocation for the data channels. It communicates with the host system and the driver stack via the Slave Interface block. One Active Manager is required per SLIMbus system.

2.3.3.3 Interface Device

Each SLIMbus component requires one Interface Device, which provides SLIMbus management services for the component. This block manages component reset so that a component can properly sequence its devices. In addition, it reports information about the status of the component to the other SLIMbus components.

2.3.3.4 Framer

SLIMbus requires one Framer on the bus to be active at a given time. The SLIMbus Host IP from Arasan can function as the Active Framer for the bus, or transfer this role to another Framer on the bus. When active, the Framer is responsible for booting the SLIMbus, generating the SLIMbus clock and providing the Frame Logic block with the frame synchronization symbol, frame structure and timing information, and guide channels.

2.3.3.5 Message Handler

This block routes to the Active Manager, Interface Device or Framer the relevant fields of all incoming messages extracted by the Frame Logic block. It performs the opposite function for outgoing messages.

2.3.3.6 Generic Device

This block contains up to 8 generic data port pairs which can transfer data To/From system memory through Slave DMA. For applications which require bridging to other protocols, like I2S or I2C, or for direct interface to DAC’s or ADC’s, the Generic FIFO Interface is made available. Each port pair can be configured either as data source or sink with respect to SLIMbus.

2.3.3.7 Frame Logic

This block, which implements the Frame layer, is responsible for synchronizing with framing, guide, message and data channels. During transmission, this block interleaves control and data channels into a single, serialized bit stream to be driven by the PHY onto the SLIMbus. For incoming bit streams, this block separates the control and data streams, and routes them to the message handler and the generic device respectively.

2.3.3.8 PHY

The Physical layer drives and captures the bit stream between SLIMbus components. Signaling is CMOS-like, with Non-Return-to-Zero Inverted (NRZI) coding for the data lines. Data is driven on the positive edge and read on the negative edge of the SLIMbus clock. The SLIMbus Host Controller IP provides the data and Clock Input/ Output and enable signals, which allows users to choose the appropriate bidirectional pads at the chip level.

2.4 PIN Diagram

2.4.1 AHB or Generic FIFO Interface to SLIMbus Pin Diagram

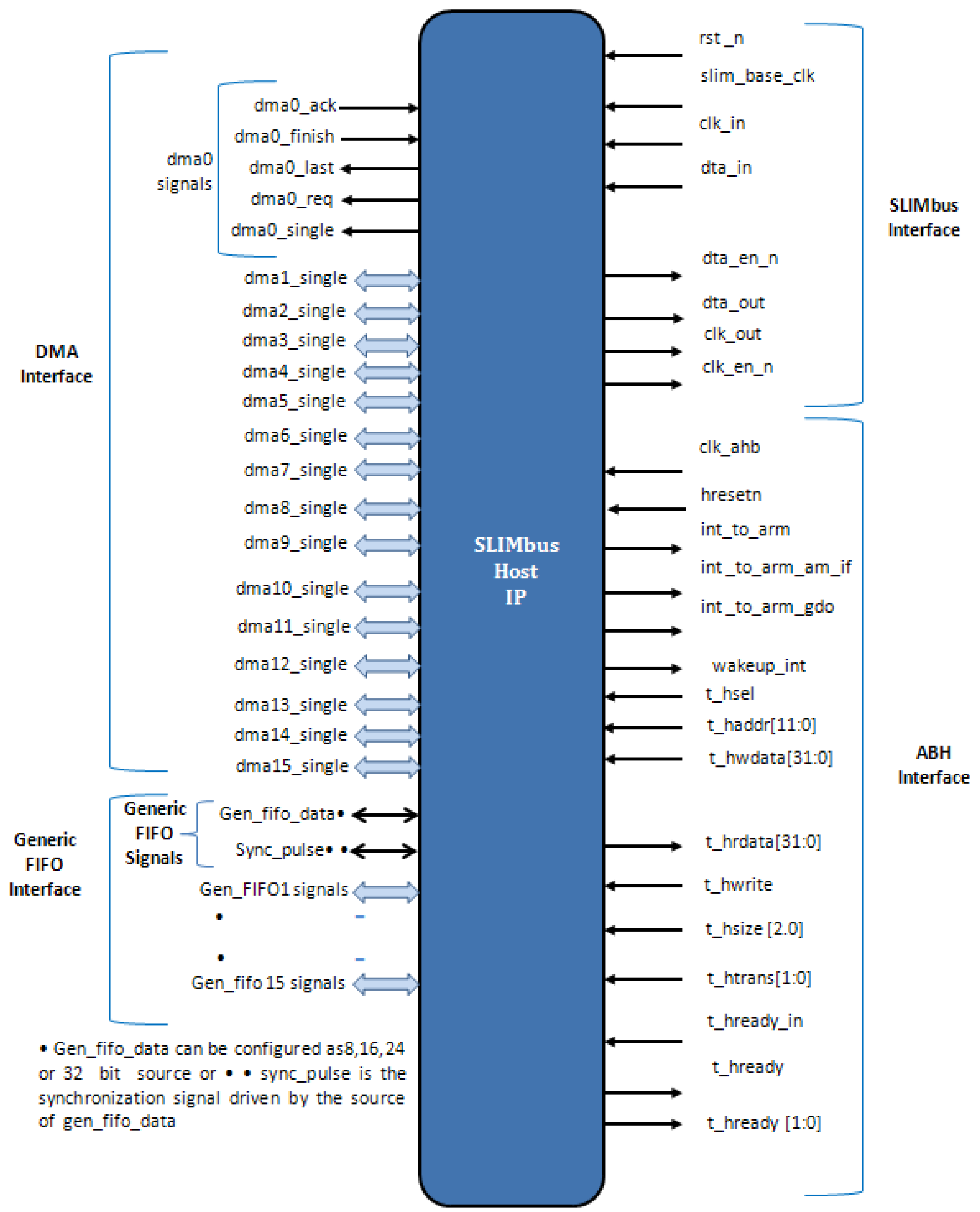

Figure 3: SLIMbus Host Controller Pinout Diagram

2.5 SoC Level Integration

2.5.1 Verification Environment

- Comprehensive suite of simulation tests for ease of SoC integration
- Verification components and test files provided
- Verification environment and test suite well documented

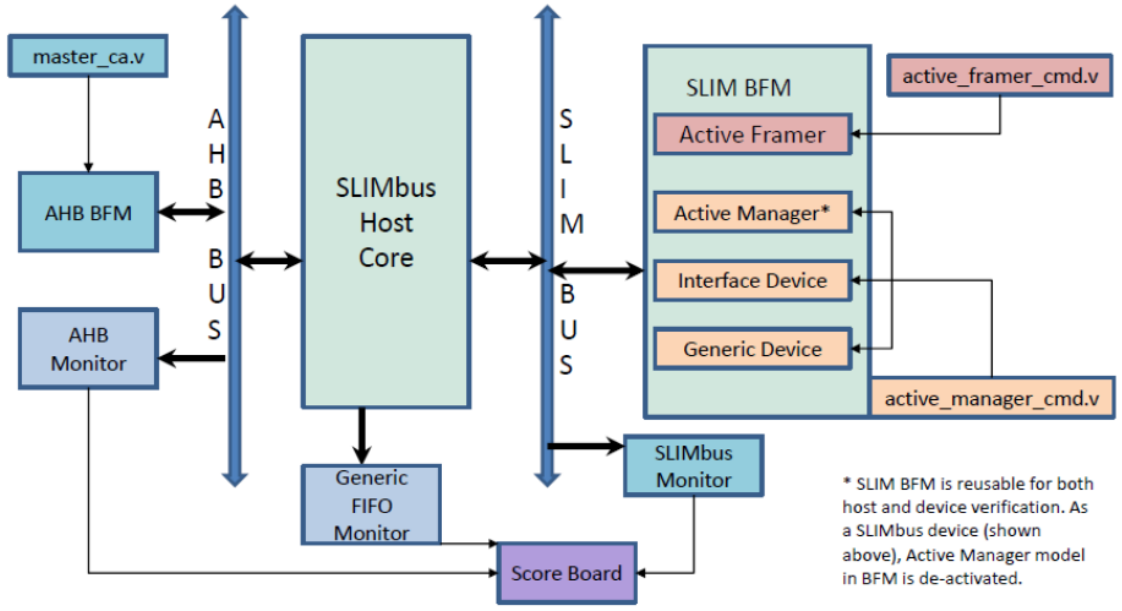

Figure 4: SLIMbus Host IP Verification Environment

2.5.2 IP Deliverables

- Verilog HDL of the IP Core
- Synthesis scripts
- Gate count estimates available upon request
- User guide
- Verification environment and tests

3 SLIMbus Device Controller IP

3.1 Overview

The Arasan SLIMbus Device Controller IP is designed to provide MIPI SLIMbus 1.1a compliant connectivity for a peripheral device, like an audio codec, to a SLIMbus compliant host, like an Applications Processor on a mobile platform, and share the bus bandwidth with other SLIMbus devices that may exist.

SLIMbus has a TDM channel allocation structure for control messages and data. When its Framer is active, the device drives the SLIMbus clock, creates the SLIMbus frames, and enables the other SLIMbus devices and the SLIMbus host to synchronize and share the available bandwidth.

SLIMbus Device IP Core contains a configurable generic device that transfers data to and from remote SLIMbus components and legacy interfaces, like I2S, I2C and SPI, or interface directly to audio DAC's and ADC's through the Generic FIFO Interface. One or more port pairs of the generic device can be used for each kind of peripheral interface, up to a maximum of 16.

At a system level, this IP operates under control of a SLIMbus host, whose active manager and associated software stack monitors the characteristics and status of the SLIMbus device, and configures its registers and access to the bus accordingly.

3.2 Features

- Compliant with MIPI SLIMbus specification version 1.1
  - Delivered in Reuse Methodology Manual (RMM) compliant Verilog RTL format
  - Small footprint

- Contains full-featured active interface device, with support for
  - All SLIMbus Core Messages
  - Error handling and recovery, as defined in the MIPI SLIMbus specification

- Embedded Framer which can be active or passive, and supports SLIMbus clock generation handoff to/from Framers in other components, with support for
  - Clock Gears 1 to 10, either as clock source or clock receiver
  - Maximum of 28.8 MHz bit-serial rate
  - Dynamic SLIMbus clock frequency scaling and clock Pause/Resume minimizes power

- Contains one generic device with up to 8 active port pairs; each port supports
  - Isochronous, Pushed, Pulled, Asynchronous Simplex, Extended Asynchronous Simplex and Locked transport protocols over SLIMbus with data segment size of 1 to 31 slots
  - Interface options to peripheral components or devices
  - Generic FIFO Interface

- Audio or peripheral data sample sizes of 8,16, 24 or 32 bits
- Different port pairs can have different sampling rates
- I2S for Audio channels
- SPI or I2C for interface to other peripherals or microcontrollers

3.3 Architecture

3.3.1 Functional Description

The Arasan SLIMbus Device Controller IP is designed to provide MIPI SLIMbus 1.1 compliant connectivity to between a peripheral device, like an audio codec, and a SLIMbus compliant SoC, like an Applications Processor on a mobile platform.

SLIMbus has a TDM channel allocation structure for control messages and data. When its Framer is active, the device drives the SLIMbus clock, creates the SLIMbus frames, and enables the other SLIMbus devices and the SLIMbus host to synchronize and share the available bandwidth.

This IP contains a configurable generic device that transfers data to and from remote SLIMbus components and legacy interfaces, like I2S, I2C and SPI, or interface directly to audio DAC's and ADC's through the Generic FIFO Interface. One or more port pairs of the generic device can be used for each kind of peripheral interface, up to a maximum of 16.

At a system level, this IP operates under control of a SLIMbus host, whose active manager and associated software stack monitors the characteristics and status of the SLIMbus device, and configures its registers and access to the bus accordingly.

3.3.2 Functional Block Diagram

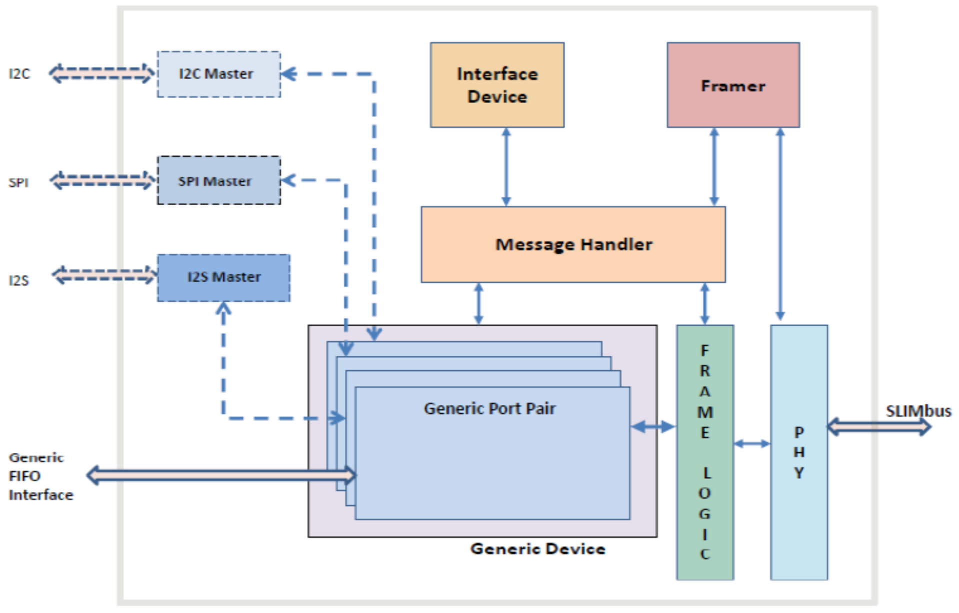

Figure 5: SLIMbus Device Functional Block Diagram

3.3.3 Functional Block Diagram Description

3.3.3.1 Interface Device

Each SLIMbus component requires one Interface Device which provides SLIMbus management services for the component. This block manages component reset so that a component can properly sequence its devices. In addition, it reports information about the status of the component to the other SLIMbus components.

3.3.3.2 Framer

SLIMbus requires one Framer on the bus to be active at a given time. The SLIMbus Host IP from Arasan can function as the Active Framer for the bus, or transfer this role to another Framer on the bus. When active, the Framer is responsible for booting the SLIMbus, generating the SLIMbus clock and providing the Frame Logic block with the frame synchronization symbol, frame structure and timing information, and guide channels.

3.3.3.3 Message Handler

This block routes to the Active Manager, Interface Device or Framer the relevant fields of all incoming messages extracted by the Frame Logic block. It performs the opposite function for outgoing messages.

3.3.3.4 Generic Device

This block contains up to 8 generic data port pairs which can transfer data to/from system memory through Slave DMA. For applications which require a direct interface to DAC's or ADC's, the Generic FIFO Interface is made available. For interface to other protocols, like I2S, I2C or SPI, the corresponding bus masters can be incorporated to perform the bridging function. These options are shown in the Functional Block Diagram. Each port pair can be configured either as data source or sink with respect to SLIMbus.

3.3.3.5 Frame Logic

This block, which implements the frame layer, is responsible for synchronizing with framing, guide, message and data channels. During transmission, this block interleaves control and data channels into a single, serialized bit stream to be driven by the PHY onto the SLIMbus. For incoming bit streams, this block separates the control and data streams, and routes them to the message handler and the generic device respectively.

3.3.3.6 PHY

The physical layer drives and captures the bit stream between SLIMbus components. Signaling is CMOS-like, with Non-Return-to-Zero Inverted (NRZI) coding for the data lines. Data is driven on the positive edge and read on the negative edge of the SLIMbus clock. The SLIMbus Host Controller IP provides the data and clock input, output and enable signals, which allows users to choose the appropriate bidirectional pads at the chip level.

3.4 PIN Diagram

3.4.1 3.1 Generic FIFO and Optional 12S/12S/SPI Interface to SLIMBus

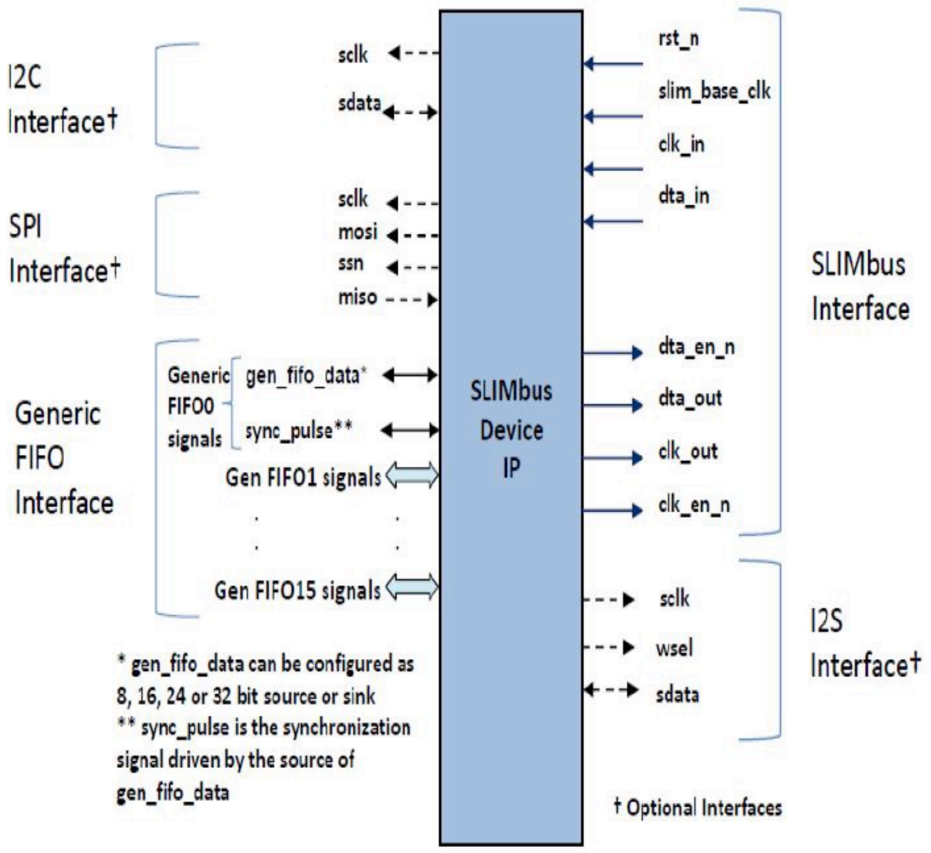

Figure 6: SLIMbus Device Pinout Diagram

3.5 SoC Level Integration

3.5.1 Verification Environment

- Comprehensive suite of simulation tests for ease of SoC integration
- Verification components and test files provided
- Verification environment and test suite well documented

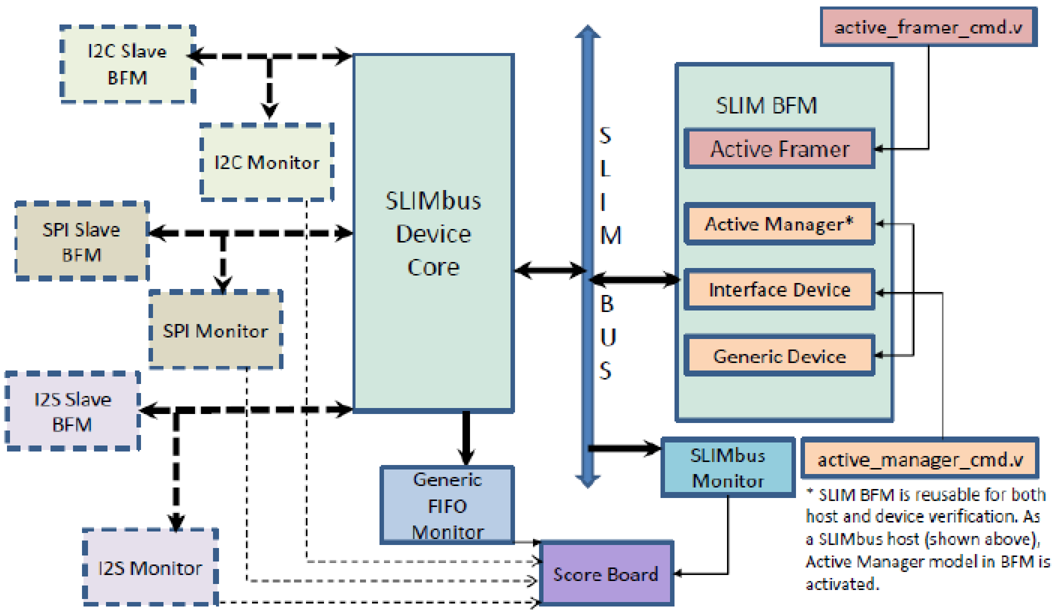

Figure 7: SLIMbus Device IP Verification Environment

3.5.2 IP Deliverables

- Verilog HDL of the IP Core
- Synthesis scripts
- Gate count estimates available upon request
- User guide
- Verification environment and tests

4 SLIMbus Hardware Validation Platform

4.1 Overview

Arasan's SLIMbus® (Serial Low-power Inter-chip Media bus protocol) Hardware Validation Platform provides the mobile industry a versatile means to assist in the development and debugging of SLIMbus® products. By emulating a real-world SLIMbus host component, this platform can be used by system and software developers to completely validate the implementation of the SLIMbus interface in their products during various stages of the development cycle.

The SLIMbus Hardware Validation Platform (HVP) includes both the Manager and Framer devices which are active by default. This gives the user the means, through a simple Exerciser application, to configure the SLIMbus sub-system, control and data space, and channel bandwidth allocation within the data space when connecting their products to the HVP. Such configuration can be modified, saved for a future runtime session, and restored in the future, thereby saving time for the user. Bus and frame management duties can be transferred to external components that contain Manager and Framer devices that the user may want to activate.

4.2 Features

- Compliant with MIPI® SLIMbus Specification version 1.1
- Configurable as SLIMbus Host Component as SLIMbus Active or Inactive Manager and SLIMbus Device Component as SLIMbus Inactive Manager
- GUI supports all possible configuration for an Active Manager or Inactive Manager as per SLIMbus specifications Version1.1
- GUI based interface to validate hardware using the software
- Supports all transport protocol for data transactions as per MIPI® SLIMbus Version1.0
- Enumerates all SLIMbus devices existing SLIMbus up to 256 data channel
- Reads SLIMbus device and port information
- Support for dynamic SLIMbus devices enumeration and get the device and port information
- Control space and data space channel can be configured dynamically
- During testing, GUI provides save, reload configurations and to edit existing configuration
- Simultaneously 256 data channel and control using SLIMbus massages (Activate/Deactivate, gear change, Connect/Disconnect, Multisink, and so on)
- Configures device parameters to clock gear
- GUI supports I2S Audio Interface with Mono, Stereo channel such as 2.1, 3.1, 4.1, 5.1, 6.1, 7.1 and audio support
- GUI support s SPI read and write using SLIMbus messages including user define messages
- Simultaneously transmits and receives data over multiple channel
- Resets devices, specific devices, and Host controller
- No protocol specific knowledge required
- Fully documented with required GUI picture, for example Test Scenario

- Premier support directly from engineering team
- Configures SLIMbus Host controller component on power-on reset and acts as active manager

4.3 Description

The SLIMbus Hardware Validation Platform (HVP) is a Linux-based system that can perform validation on SLIMbus Host/Device components. It can be used for SoC validation, early software development and for limited production testing. The HVP provides solutions that you need to launch your products in the shortest possible time including a binary FPGA implementation of Arasan's market leading SLIMbus Host component IP or SLIMbus Device component IP and software drivers running on a Linux OS which enables user-written programs to fully utilize the controller functions. The Arasan HVP comprises a PC platform which can run in the Fedora Linux operating system. An FPGA IP board is pre-programmed with SLIMbus Host component IP which contains SLIMbus Manager, Interface, Framer and Generic device. The DUT can either be another HVP with a complementary IP configuration or a SoC containing a SLIMbus Host component or SLIMbus Device component. The SLIMbus HVP also includes a binary version of Arasan's SLIMbus software stack. The software stack can also be used for validating SLIMbus Host/Device components during its development and integration into life cycles thereby helping designers to reduce the time to market their product.

The SLIMbus protocol specific driver stack layer implements protocols such that it can handle all SLIMbus messages and can control data transfer between different SLIMbus Components (SLIMbus Host Component to and from SLIMbus Device Component or any SLIMbus Device Component to and from any other SLIMbus Device Component) present over SLIMbus. Using Stack, we can control Active Manager of Arasan SLIMbus Host Component and can control Inactive Manager, if already any other existing Active Manager, controlling SLIMbus.

As an Active Manager, it will perform all message handling, enumerate all the devices present over SLIMbus, configure the data channel with all kinds of data transfer protocol support by SLIMbus and also can configure and control different SLIMbus component using messages.

It supports data transfer using multiple channel between different SLIMbus devices using pushed, pulled, locked, isochronous and asynchronous protocols. It also supports functionality specific to the Arasan SLIMbus Host Component and SLIMbus Device Component IP cores. SLIMbus hardware specific driver layer is a hardware dependent layer. The layered architecture allows porting to various operating systems, various platforms and various SLIMbus® hardware devices.

4.4 Architecture

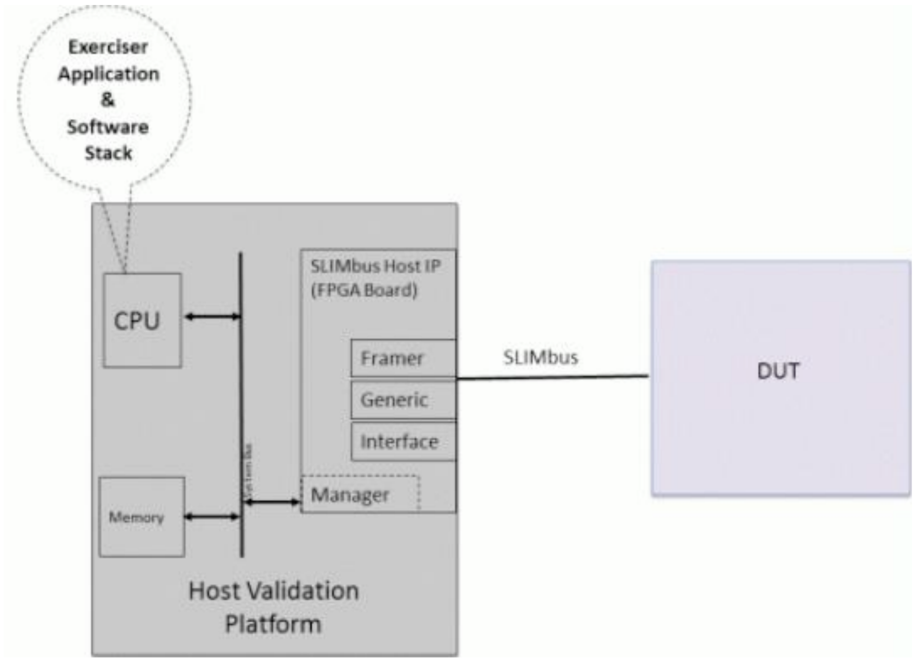

Figure 8: SLIMbus HVP Architecture

5 SLIMbus Stack

5.1 Overview

Arasan's Serial Low-Power Inter-chip Media bus protocol (SLIMbus®) software stack provides developers a method for easy development, integration, and validation of system software. The SLIMbus software stack is operating system/processor agnostic and provides a generic set of APIs to the functional driver which abstracts the SLIM protocol specific functionality. The software stack supports a variety of function drivers such as SPI, UART, BT, I2S, I2C, DAC, ADC, and Flash which make use of this generic API set to communicate on the common bus. APIs include commonly used device operations such as initialization, device configuration, data transfer, power management, and registration of call back for interrupt handling.

The SLIMbus software stack provides generic APIs to the function driver and abstracts the SLIM protocol specific functionality. The software stack supports a variety of functional drivers such as SPI, UART, BT, I2S, I2C, DAC, ADC and Flash which make use of this generic API set to communicate on a common bus. The APIs include commonly used device operations such as initialization, device configuration, data transfer, power management, and registration of call back for interrupt handling.

The SLIMbus protocol specific driver layer implements protocols that can handle all SLIMbus messages and can control data transfer between different SLIMbus components present on SLIMbus Interface.

Using the software stack, we can control an Active Manager of Arasan's SLIMbus Host Component and can control Non -Active Manager if they exist in system if another Active Manager is controlling SLIMbus. As an Active Manager the software is responsible for all message handling, device enumerate all the devices presents over SLIMbus, configure the data channel per the specification and also can configure and control different SLIMbus component using messages. It supports data transfer using multiple channel between different SLIMbus device using pushed, pulled, locked, isochronous and it also supports functionality specific to the Arasan SLIMbus Host Component and SLIMbus Device Component IP cores. SLIMbus hardware specific driver layer is dependent on the devices implemented. The layered architecture allows porting to any operating system, platforms and various SLIMbus hardware devices.

5.2 Features

- Compliant with MIPI SLIMbus® specification version 1.1
- Portability in choice of OS, processors and hardware
- Easy-to-use API interface for applications
- Fully documented generic interface API
- Easy functional driver integration to SLIMbus interface
- No protocol specific knowledge required

5.3 Architecture

5.3.1 Description

The MIPI Alliance, working towards consistency in processor and peripheral interfaces promoting reuse, interoperability, and compatibility in mobile applications. The MIPI Alliance developed and ratified the SLIMbus specification which provides a standard, robust, scalable, low-power, high-speed, cost-effective, two wire multi-drop interface that supports a wide range of digital audio and control solutions for mobile terminals. SLIMbus is an efficient bus for devices such as speakers, ADC, DAC, and Bluetooth.

The Arasan SLIMbus software supports SLIMbus systems running on many operating systems and hardware platforms. The software supports systems with single SLIMbus Host component and multiple SLIMbus Device components. The SLIMbus software supports devices that conform to MIPI SLIMbus version 1.1 specification and supports Linux and other operation systems. It can also be ported to different hardware platforms.

The Arasan SLIMbus software stack consists of three layers: General Application Interface layer (API layer), SLIMbus protocol specific driver layer and SLIMbus hardware specific driver layer. Client applications interface with the General API layer directly or through a Device Class Driver layer. The SLIMbus protocol specific driver layer implements protocols such as isochronous and asynchronous data transfers, device enumeration, and message handling. It also supports functionalities specific to the Arasan SLIMbus Host and SLIMbus Device IP cores. SLIMbus hardware specific driver layer is a hardware dependent layer. The layered architecture allows porting to various operating systems, various platforms and various SLIMbus hardware devices.

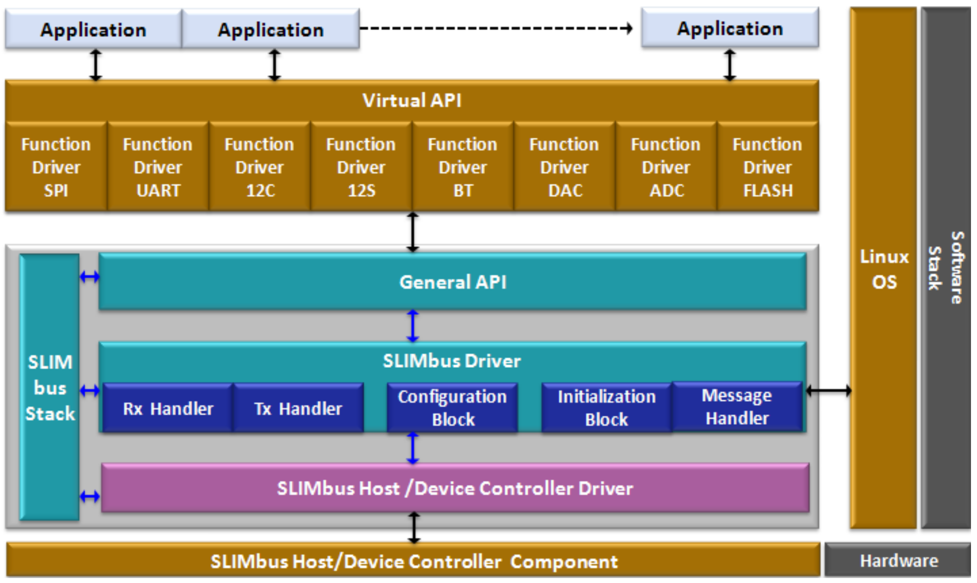

Figure 9:SLIMbus Stack Architecture

5.4 Deliverables

- Source Code and/or binaries for SLIMbus Stack
- User Manual
- API Guide
- Release Notes

6 MIPI SoundWire Master Controller

6.1 Overview

This document describes the Arasan IP Core that functions as a MIPI SoundWire® Master Controller, typically integrated into application processor or audio DSP used in smart phones, tablets and mobile PCs. The IP when integrated provides SoundWire, a new audio interface to connect to audio peripherals such as PCM or PDM amplifiers, microphones and audio codecs.

6.2 Features

- Compliant with latest draft MIPI SoundWire specification version 1.1
  - Delivered in Synthesizable Verilog RTL format
- Configurable number of Data Ports
- Flexible Port Direction
  - Source or Sink or run time configurable
- Implements clock gearbox with programmable frequency divider
- Implements SoundWire Bus Clock Stop and WakeUp detection
- Implements High-PHY support
- Implements all three layers and provides a simple application interface
  - PHY
  - Framer
  - Transport (Data and Control Ports)
  - Application Interface

- PHY
  - Modified NRZI Data coding
  - Bus clash detection

- Framer
  - Configurable frame shape
    - Supports on-the-fly changing
  - Clock Stop & Start
    - Detects wake-up event from slaves and restarts clock
  - Frame Sync generation (static & dynamic)
  - Automatic PING generation or on demand from CPU
  - Automatic SSP generation at CPU programmed interval
  - Multiplexing/De-multiplexing of payload from multiple Data ports
  - Command(Read/Write generation)
  - Parity generation & checking
  - Supports BREQ & BREL

- Transport (Data and Control Ports)
  - Configurable number of Data ports
    - Maximum of 165 ports possible (11 devices * 15 ports each)
  - Supports all the standard data port registers
    - Implements Data Port Registers(DPN_*) for each port
    - Programmed/accessed via AHB slave interface
  - Implements Control Port (MCP) registers
  - Supports Test modes (Static_1/Static_0/PRBS)

- Application/Client Interface
  - DATA Port(s) Interface
    - Channel Prepare
    - Timing reference related to Sample Event
    - Simple FIFO interface to pass audio samples from/to application for each Data port
    - Configurable direction(Source or Sink)
  - AHB slave Interface
    - Access to internal Registers (MCP, DATA Port as well as implementation specific)
    - Command & Data registers
  - For CPU to execute SoundWire commands (Read/Write/Ping) to external slaves
  - Interrupt line
    - Implements Interrupt Hierarchy and Cascades recommended in the SoundWire Specification

6.3 Architecture

6.3.1 Functional Description

The SoundWire Master IP has been architected and designed to simplify the job of implementing SoundWire connectivity in a mobile platform easy. The IP is typically integrated into application processors or audio subsystem and depending on mobile system audio architecture, connects to various low power audio devices such as speakers, microphones, earpieces etc. The IP hides all the complex framing and protocol and provides a simple easy to understand and easy to integrate application interface. It implements all the non-application dependent physical and protocol functions as defined in the MIPI SoundWire specification. It has layered implementation containing PHY, Framing and Transport. It provides simple FIFO interface for each Data port transferring audio stream samples between application layer and SoundWire. The design is flexible and highly parameterized to realize only what is needed in a given application. Typical configuration parameters include number of Data ports, number of Lanes, sample FIFO depths etc.

6.3.2 Functional Block Diagram

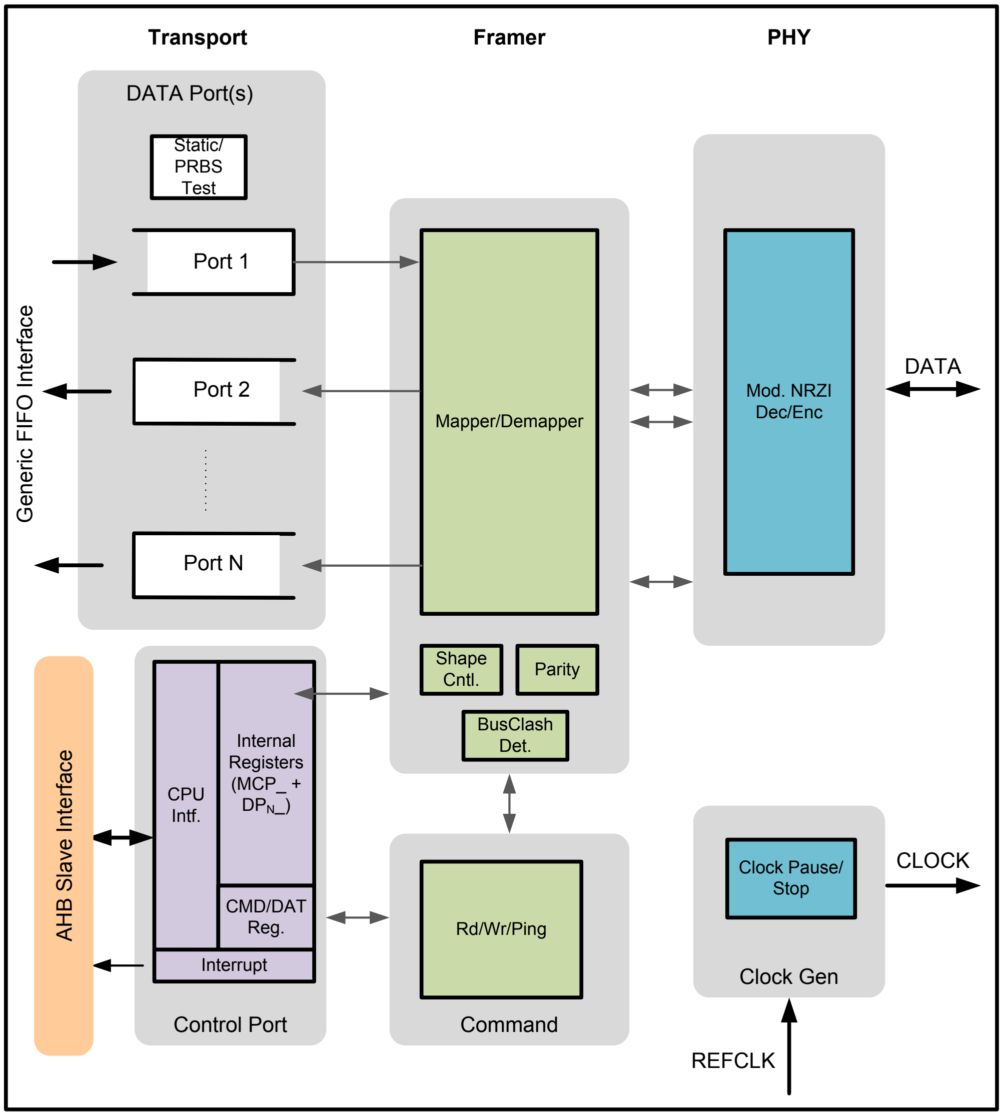

Figure 10: SoundWire Master Functional Block Diagram

6.3.3 Functional Block Diagram Description

The following sub-sections describe the functional behavior of all the major layers.

6.3.3.1 PHY

The physical layer block implements all the line-side functions such as NRZI encoding & decoding, bus clash detection, data line buffer enable/disable.

6.3.3.2 Framer

The framing layer handles the protocol related to framing and data processing. On the line-side, it has the Frame Generator that outputs the SoundWire frames of configurable shape. The Sync. Generator creates and maps Static and Dynamic synchronization vectors onto the Control column to be used by SoundWire Slaves to synchronize and lock to SoundWire frames.

Once initial configuration is done and frames start flowing, it multilplexes and maps audio payload samples received from application via Data ports onto SoundWire frames as per the controls provisioned by the Master via Data Port registers. Similarly, it performs the opposite de-mapping function for the samples received over SoundWire and de-multiplexes them into the corresponding Data port FIFOs. The Data port registers define the shape and size of Transport Sub-Frames and their multiplexing/de-multiplexing onto common Soundwire frame that may be shared by multiple devices and multiple Data ports within each device. In addition, control payload is mapped/de-mapped onto Column 0 of first 48 rows dedicated for the purpose. This includes SoundWire Slave Register read/write commands and read responses, and Ping command as well as device status generated by the Slaves in response.

6.3.3.3 Transport

The Data Port of the transport layer implements a FIFO for each Data port. The FIFO serves as small holding buffer for samples waiting to be mapped onto SoundWire Transport Sub-Frames outbound, or waiting to be passed onto application logic inbound. The width, depth and direction are configurable; the width typically matches sample size (WordLength), and the depth depends on the latency desired.

The Control Port of the transport layer implements the standard Registers defined in the MIPI SoundWire Specification that application can access and program. These registers are accessed by the application via back-door over APBB slave interface.

6.3.3.4 Application Interface

In the Data path, the Application Interface provides a simple and quick push/pop style FIFO interface for each Data port for samples. The application logic just pushes the samples into the FIFO, one at a time as long as there is room in the FIFO indicated by ready or not full signal in the outbound direction. The IP pops and transfers the samples over to the application logic on the inbound direction whenever FIFO is not empty. In the Control path, the Application Interface provides an APB slave interface that provides a back-door access to the Registers.

6.4 PIN Diagram

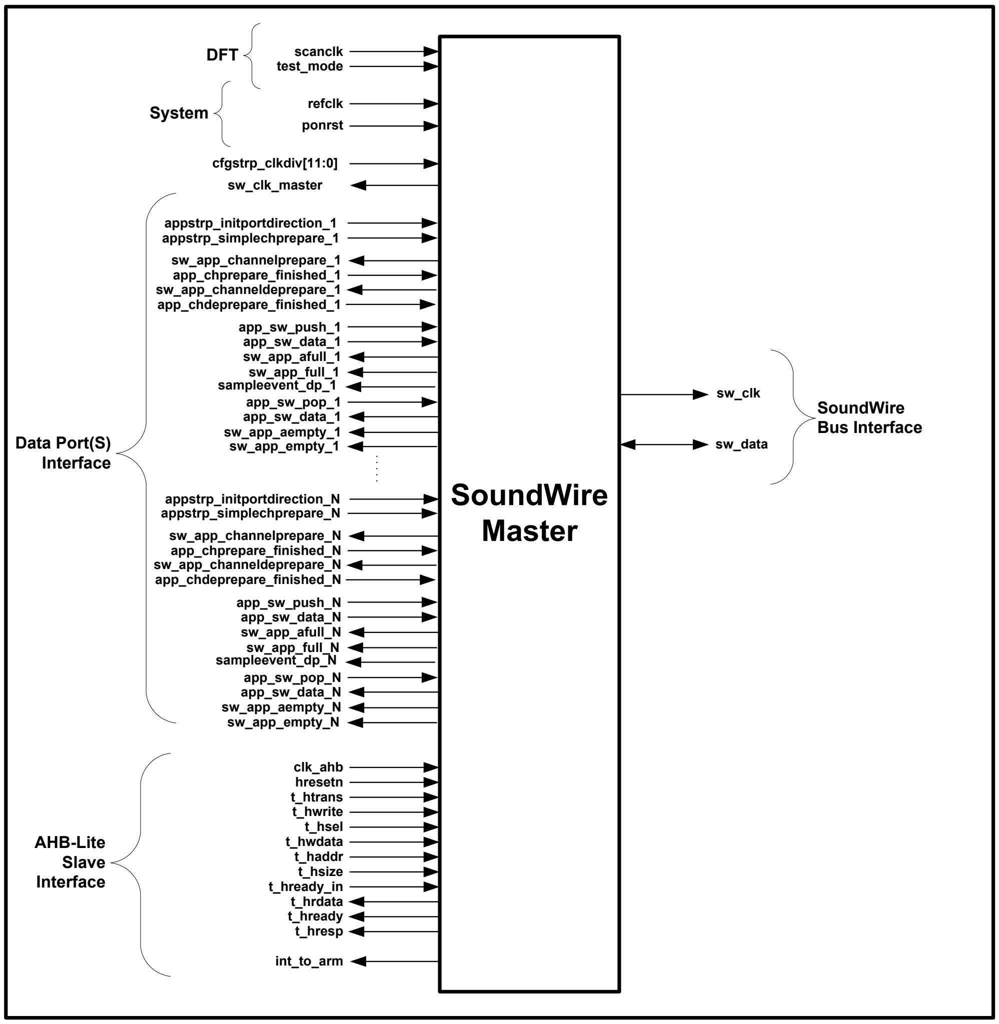

Figure 11: SoundWire Master Pinout Diagram

6.5 SoC Level Integration

6.5.1 Verification Environment

The SoundWire Master IP Design Under Test (DUT) is written in synthesizable Verilog. On the processor side it has two interfaces; (a) DATA Port Interface, (b) Contol Intreface. The DATA Port Interface presents a simple FIFO for each DATA Port. The FIFO interface is used for transporting audio payloads. The Control Interface implements a simple SRAM style acess mechanism with Chip Select. The Control Interface is used by the AHB Processor to both (a) Configure Control and Data Port registers associated with Master IP itself; (b) Pass Register Read/Write commands needed to enumerate and configure SoundWire Slave devices.

These verification environment comes the SoundWire Slave BFM that emulates generic slave behavior and has error reporting and injection capabilities. It is implemented in System Verilog. The slave BFM implements the standard Slave Control Port registers and can emulate configurable number of DATA Ports and associated Registers. In addition, it has hooks that connect to generic DATA Port Sink and Source modules that mimic SoundWire DATA Ports.

The DATA Port Traffic Generator on the SoC side as well as DATA Port Source and Sink on the Slave BFM side report the payloads sent and received to DATA Intergrity Checker/Scoreboard that compares the data and displays mismatch warning/error messages appropriately.

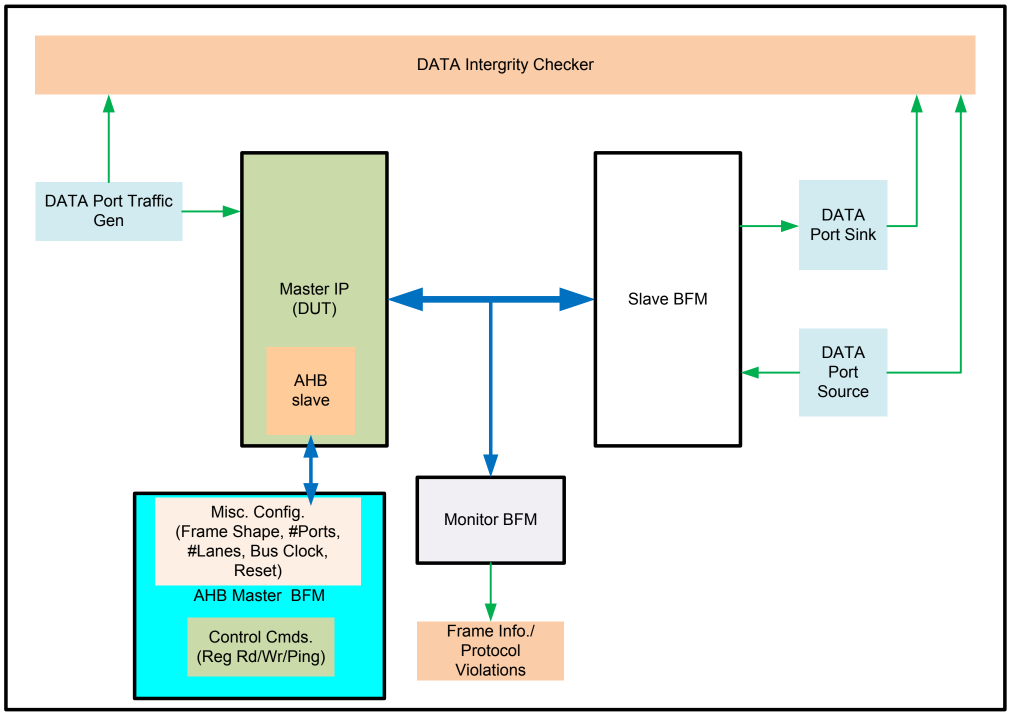

Figure 12: SoundWire Master Controller IP Verification Environment

6.5.2 Verification Deliverables

- Comprehensive suite of simulation tests for ease of SoC integration
- Verification components and test files provided
- Verification environment and test suite well documented

6.5.3 IP Deliverables

- Verilog HDL of the IP Core
- Synthesis scripts
- Gate count estimates available upon request
- User guide
- Verification environment and tests

7 SoundWire Slave Controller

7.1 Overview

Arasan Chip Systems is a leading SoC IP provider of a complete suite of MIPI® compliant IP solutions which consist of analog PHY and digital controller IP cores, verification IP, software stacks and drivers, hardware validation platforms for software development and compliance testing, and optional customization services.

The MIPI compliant IP cores serve as interface building blocks that simplify sub-system level interconnect architectures in mobile platforms. This leads to smaller footprint, greater interoperability between mobile IP, chips and devices from diverse sources, and lower power and EMI.

This document describes the Arasan IP Core that functions as a MIPI SoundWire® Slave Controller, typically integrated into PCM or PDM amplifiers, microphones and audio codecs used in smart phones, tablets, mobile PCs. The IP provides SoundWire, a new audio interface to connect to master normally residing in the application processor or audio co-processor/DSP.

7.2 Features

- Compliant with MIPI SoundWire specification version1.1
    - Delivered in Synthesizable Verilog RTL format
    - Small footprint
- Configurable number of Data Ports
- Configurable Direction
    - Source or Sink or run time configurable
- Support for Reduced Data Port
    - Provides flexibility over Simplified Data Port
- BRA Support over Data Port 0 for faster firmware download
- Supports full address range using Address Paging registers
- Supports accesses to registers implemented external to the IP
- Supports ClockStop Modes
- Implements High PHY support
- Implements all three layers and provides a simple application interface
    - PHY
    - Framer
    - Transport (Data and Control Ports)
    - Application Interface
- PHY

    - Modified NRZI Data coding
    - Bus clash detection

- Framer
    - Supports all frame shapes defined in the Spec.
        - Auto synchronizes to incoming frame shape.
    - Frame Sync detection & checking (static & dynamic)
    - SSP detection
    - Parity checking
    - Multiplexing/De-multiplexing of payload from multiple Data ports
    - Command(Read/Write/PING) decode and response
    - Generates timing reference to application layer for each DP based on SSP event and programmed SampleInterval

- Transport (Data and Control Ports)
    - Configurable number of Data ports
        - Maximum of 14 ports possible, each port with up to 8 audio channels
    - Supports all the standard data port registers
        - Implements Data Port Registers(DP_N_*) for each port
        - Programmed/accessed by Master via SoundWire bus
    - Implements Slave Control Port (SCP) registers
    - Supports Test modes (Static_1/Static_0/PRBS)
    - BRA over Data Port 0

- Application/Client Interface
    - DATA Port(s) Interface
        - Channel Prepare
        - Timing reference related to Sample Event
        - Simple FIFO interface to pass audio samples from/to application for each Data port
        - Configurable direction(Source or Sink)
    - BRA Port
        - Block transfer over Data Port 0
        - Supports both Read and Write transfers
    - External Register Interface
        - Supports accesses to external registers

7.3 Architecture

7.3.1 Functional Description

- The SoundWire Slave IP has been architected and designed to simplify the job of adding SoundWire interface to various low power mobile audio devices such as amplifiers, speakers and microphones. The IP hides all the complex framing and protocol and provides a simple easy to understand and easy to integrate application interface. It implements all the non-application dependent physical and protocol functions as defined in the MIPI SoundWire specification. It has layered implementation containing PHY, Framing and Transport. It provides simple FIFO interface for each Data port transferring audio stream samples between application layer and SoundWire. The design is flexible and highly parameterized to realize only what is needed in a given application. Typical configuration parameters include number of Data ports, number of Lanes, sample FIFO depths etc.

7.3.2 Functional Block Diagram

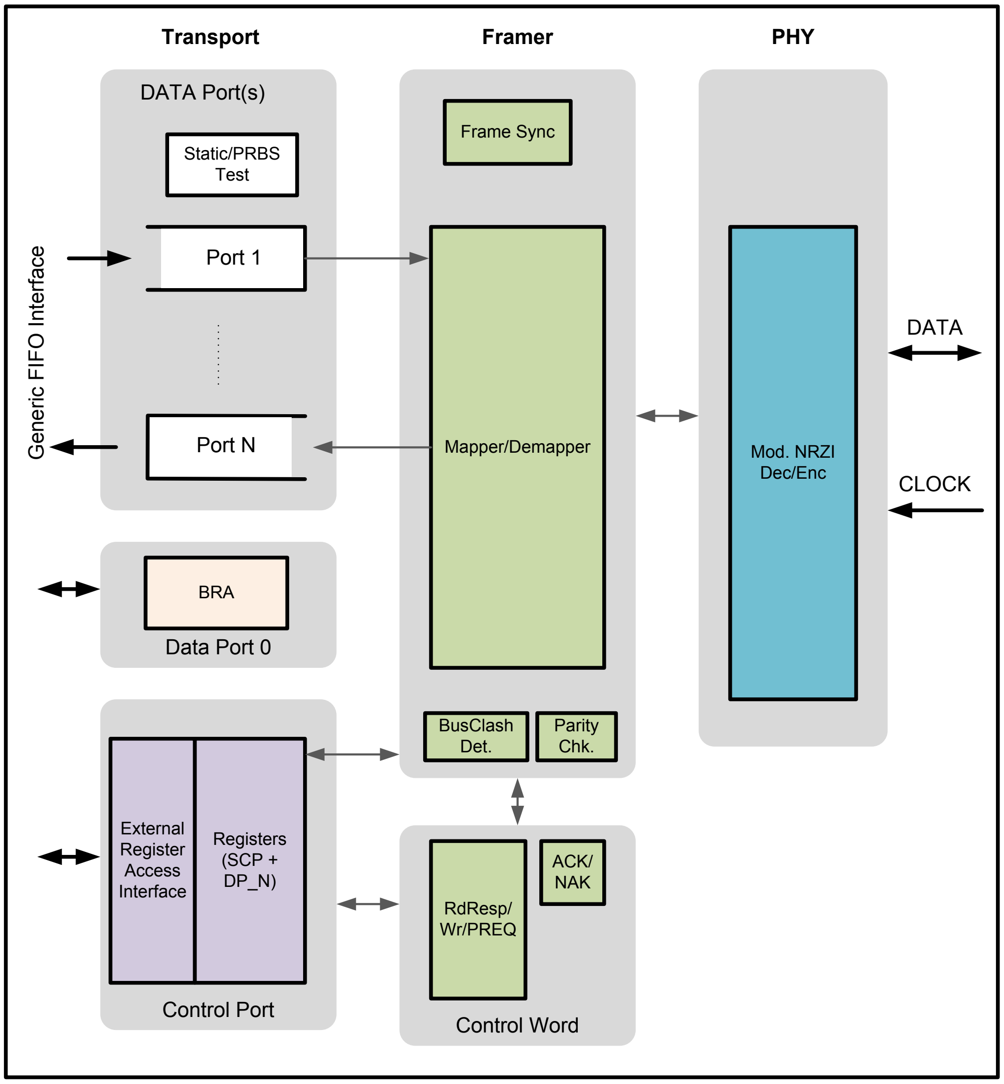

Figure 13: SoundWire Slave Functional Block Diagram

7.3.3 Functional Block Diagram Description

The following sub-sections describe the functional behavior of all the major layers.

7.3.3.1 PHY

The physical layer block implements all the line-side functions such as NRZI encoding & decoding, bus clash detection, data line buffer enable/disable.

7.3.3.2 Framer

The framing layer handles the protocol related to framing and data processing. On the line-side, it has the Frame Synchronizer that hunts for static and dynamic synchronization patterns and synchronizes itself to incoming frames and becomes itself attached. In addition, framer decodes and interprets the PING and register READ, WRITE commands from the Master and responds appropriately. The framer also detects parity errors and bus clashes.

Once initial configuration is done and frames start flowing, it multiplexes and maps audio payload samples received from application via Data ports onto SoundWire frames as per the controls provisioned by the Master via Data Port registers. Similarly, it performs the opposite de-mapping function to the samples received over SoundWire and de-multiplexes them into the corresponding Data port FIFOs. The Data port registers define the shape and size of Transport Sub-Frames and their multiplexing/de-multiplexing onto common Soundwire frame that may be shared by multiple devices and multiple Data ports within each device.

7.3.3.3 Transport

- **Audio Data Ports**

The Data Port(s) block of the transport layer implements a FIFO for each Data port. The FIFO serves as small holding buffer for samples waiting to be mapped onto SoundWire Transport Sub-Frames outbound, or waiting to be passed onto application logic inbound. The width is configured to match one sample frame. The depth and direction are configurable as well.

The Control Port of the transport layer implements the standard SCP and DP_ Registers defined in the Specification that Master can access and program.

- **Bulk Register Access Data Port**

The Slave controller IP implements Data Port 0 and related registers for Bulk Register Access (BRA). The BRA is convenient for bulk register access such as software image download. The IP implements BRA Target function and supports both write and read operations. The SoundWire Master pushes a block of data to Slave in a write operation, the SoundWrite Master pulls a block of data from the Slave in a read operation. The IP uses a dedicated interface for passing data from/to application/user layer. Note that BRA accesses are always passed to the application layer, IP provides no means to access internal SCP or Data Port registers using BRA.

7.3.3.4 Application Interface

- **Audio Data Ports**

In the Data path, the Application Interface provides a simple push(Source DATA Port) or pop(Sink Data Port) style FIFO interface for each Data port for audio samples. For Source DATA Port, the application logic just pushes the samples into the FIFO, one Sample Frame, i.e., set of samples, one from each enabled channel at a time as long as there is room in the FIFO indicated by ready or not full signal in the outbound direction. Similarly, for Sink DATA Ports, application pops audio samples, one Sample Frame at a time whenever FIFO is ready or not empty. The IP outputs a one clock wide Sample Event strobe for each DATA Port, that may be used by application as timing event for pushing & popping samples. The Sample Event strobe is generated at the beginning of each Sample Interval or at a system-wide SSP Event.

- **BRA Port**

The BRA Port uses a dedicated SRAM style interface for passing BRA blocks from/to the Application. Note that IP only supports BRA over DP0, meaning a value of b00 in DP0_PortCtrl.BPT_PayloadType. It doesn't support other values.

7.4 PIN Diagram

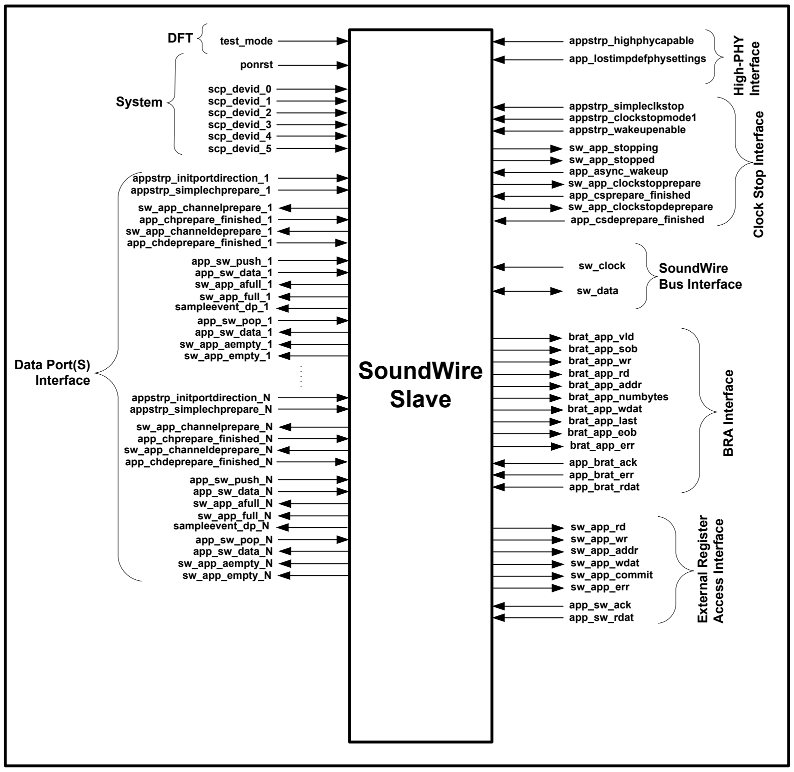

Figure 14: SoundWire Slave Pinout Diagram

7.5 SoC Level Integration

7.5.1 Verification Environment

This section provides information about the architecture of the SoundWire Slave IP Verification Environment.

The SoundWire Slave IP Design Under Test (DUT) is written in synthesizable Verilog. On the application side it has one interface; DATA Port Interface. The DATA Port Interface presents a simple FIFO for each DATA Port. The FIFO interface is used for transporting audio payloads. The SoundWire slaves are configured and managed completely by SoundWire Master and all commands are passed over the SoundWire bus.

These verification environment comes with the SoundWire Master BFM that emulates generic master behavior and has error reporting and injection capabilities. It is implemented in Verilog. The master BFM implements the standard Master Control Port registers and can emulate configurable number of DATA Ports and associated Registers. In addition, it has hooks that connect to generic DATA Port Sink and Source modules that mimic SoundWire DATA Ports.

The DATA Port Traffic Generator on the application side as well as DATA Port Source and Sink on the Slave BFM side report the payloads sent and received to DATA Intergrity Checker/Scoreboard that compares the data and displays mismatch warning/error messages appropriately.

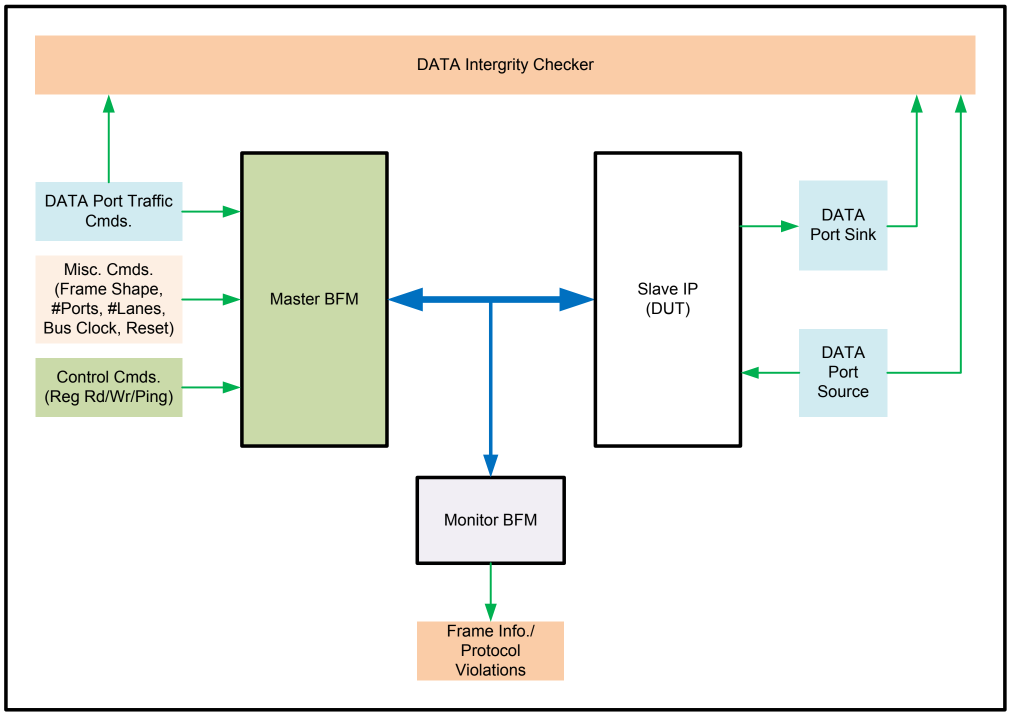
DATA Intergrity Checker
DATA Port Traffic Cmds.
Misc. Cmds. (Frame Shape, #Ports, #Lanes, Bus Clock, Reset)
Control Cmds. (Reg Rd/Wr/Ping)
Master BFM
Slave IP (DUT)
DATA Port Sink
DATA Port Source
Monitor BFM
Frame Info./ Protocol Violations

Figure 15: SoundWire Slave Controller IP Verification Environment

7.5.2 Verification Deliverables

- Comprehensive suite of simulation tests for ease of SoC integration
- Verification components and test files provided
- Verification environment and test suite well documented

7.5.3 IP Deliverables

- Verilog HDL of the IP Core
- Synthesis scripts
- Gate count estimates available upon request
- User guide
- Verification environment and tests

8 SoundWire Hardware Validation Platform

8.1 Overview

Arasan's SoundWire Hardware Validation Platform provides the mobile industry a means to assist in the development and debugging of SoundWire products. By emulating a real-world SoundWire master component, this platform can be used by system and software developers to completely validate the implementation of the SoundWire interface in their products during various stages of the development cycle.

The SoundWire Hardware Validation Platform (HVP) gives the user the means, through a simple console application, to configure the SoundWire sub-system, control and data parameters, and channel/port allocation within the data space when connecting their products to the HVP. Such configuration can be modified whenever required from the application.

8.2 Features

- Compliant with latest draft MIPI SoundWire specification version 1.0
- GUI based interface to validate hardware using the software
- Enumerates all SoundWireslaves existing in SoundWire bus
- Reads SoundWireslaves' Device ID information saying Part ID, Class ID, Manufacturer ID, version number etc.,
- Control and data ports can be configured dynamically
- Slaves can be configured specifically by accessing slave control ports
- Master can be configured by accessing master control ports
- Reports slave status as and when there is a change in status.
- Provides a means to know occupied/free ports thereby allowing users to configure free ports next time
- Frame shapes can be changed during port data transfer
- Provides a means to reset/start/stop clocks
- Allows Bank switching thereby allows the users to start the next transfer with new channel configuration

8.3 Description

The SoundWire Hardware Validation Platform (HVP) is a Linux-based system that can perform validation on SoundWireMaster/Slave components. It can be used for SoC validation, early software development and for limited production testing. The HVP provides solutions that you need to launch your products in the shortest possible time including a binary FPGA implementation of Arasan's market leading SoundWire Master component IP or SoundWireSlave component IP and software drivers running on a Linux OS which enables user-written programs to fully utilize the controller functions. The Arasan HVP comprises a PC platform which can run in the Fedora Linux operating system. An FPGA IP board is pre-programmed with SoundWireMaster component IP which in turn communicates with any SoundWire Slave component IP. The DUT can either beanother HVP with a complementary IP configuration or a SoC containing a SoundWireMaster component or SoundWireSlave component. The SoundWire HVP also includes a version of Arasan's SoundWire software stack. The software stack can also be used for validating SoundWireMaster/Slave components during its development and integration into life cycles thereby helping designers to reduce the time to market their product.

The SoundWire protocol specific driver stack layer implements protocols such that it can handle all SoundWire configurations and can control data transfer between different SoundWire Components present over SoundWire. It supports data transfer using multiple channel/ports between different SoundWirecomponents.The software stack has OS wrappers layer which can be easily ported to any OS.

8.4 Architecture

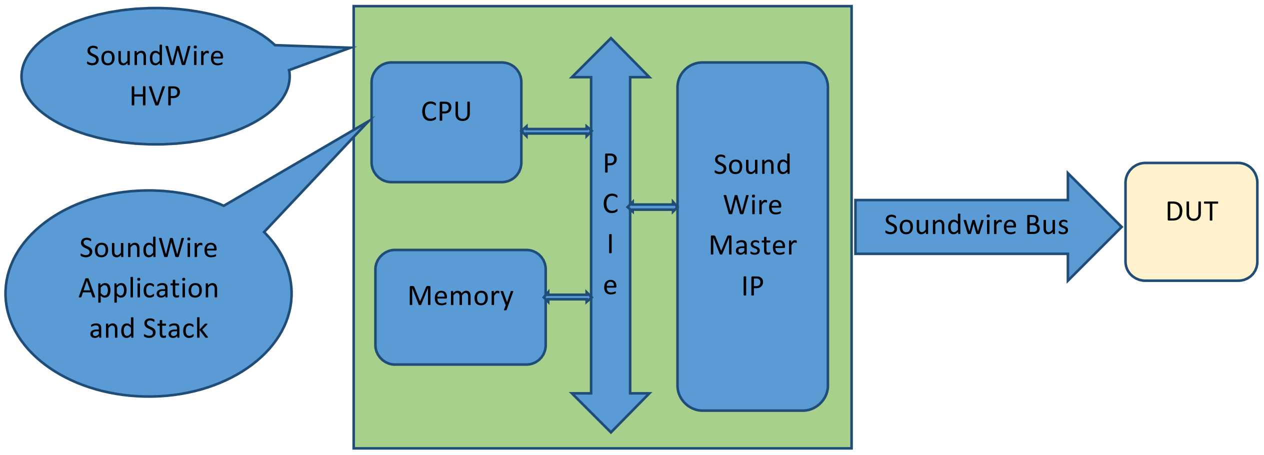

Figure 16: SoundWire HVP Architecture

8.5 Deliverables

- Controller IP implemented in FPGA
- SoundWireSoftwareStack
- LinuxInstallerPackagewith Documentation
- Documentation

9 SoundWire Stack

9.1 Overview

Arasan's SoundWire software stack provides developers a method for easy development, integration, and validation of system software. The software stack is operating system/processor agnostic and provides a generic set of APIs to the functional driver which abstracts the SoundWire protocol specific functionality. APIs include commonly used device operations such as enumeration, master/slave control/port configuration, port data transfer, reset, and registration of call back for interrupt handling.

The software stack has a layered architecture which allows the users to understand or update the protocol functionalities easily.

The SoundWire stack provides set of APIs which can be used to control the SoundWire Master and Slaves. The GUI application lists out the features supported in the stack which allows the users to configure master and slaves.

The software stack has OS wrappers layer which can be easily ported to any operating system or any bus interface.

9.2 Features

- Compliant with latest draft MIPI SoundWire specification version 1.0
- Portability in choice of OS, processors and hardware
- Easy-to-use API interface for applications
- Fully documented generic interface API
- Easy functional driver integration to SoundWire interface
- No protocol specific knowledge required

9.3 Architecture

9.3.1 Description

The Arasan SoundWire software supports SoundWire systems running on many operating systems and hardware platforms. The software supports systems with single SoundWireMaster component and multiple SoundWireslave components. The SoundWire software supports any custom Soundwire slaves as it is being controlled via the Soundwire Master component. It can also be ported to different hardware platforms.

The Arasan SoundWire software stack consists of three layers: General Application Interface layer (API layer), SoundWire protocol specific driver layer and SoundWire hardware specific driver layer. Client applications interface with the General API layer directly or through a Device Class Driver layer. The SoundWire protocol specific driver layer implements protocols such as port data transfers, device enumeration, and bus configuration. SoundWire hardware specific driver layer is a hardware dependent layer. The layered architecture allows porting to various operating systems, various platforms and various SoundWire hardware devices.

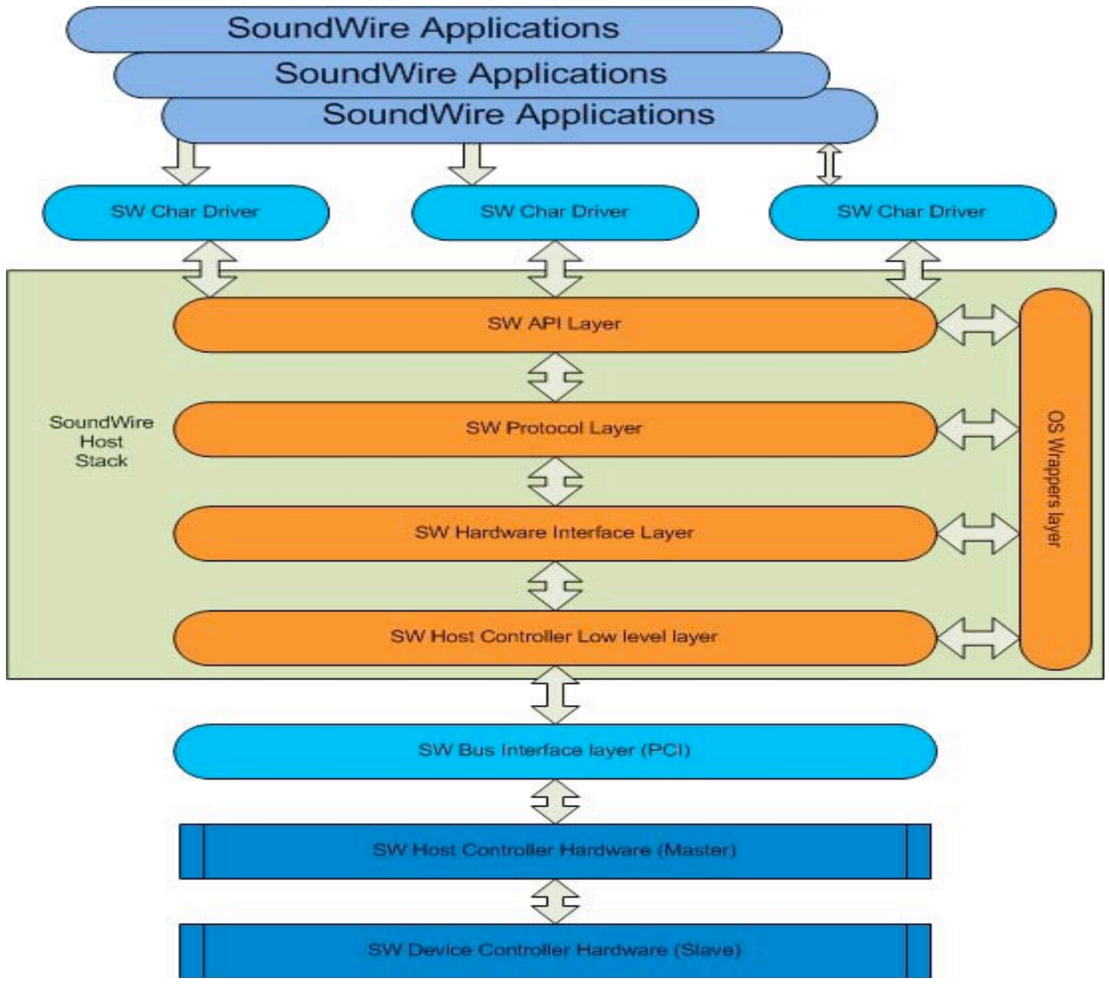

Figure 17:SoundWire Stack Architecture

9.4 Deliverables

- Source Code and/or binaries for SLIMbus Stack
- API Guide
- Release Notes

10 I2S Controller

10.1 Overview

The Arasan I2S Controller IP Core is a two-channel I2S serial audio controller compliant to the Philips* Inter-IC Sound specification. The I2S bus is used for connecting audio components such as speakers, DACs, or audio subsystems.

The Arasan I2S Controller can simultaneously send and receive data through the I2S bus. This simultaneous operation is provided by the presence of two separate FIFOs for transmission and reception. The transmit FIFO and receive FIFO handle data transfers between the I2S interface and application interface. These two interfaces can be operated in two independent clock domains. The I2S Controller also includes interrupt support for reporting FIFO and other conditions. The I2S transmitter and receiver can be configured for both I2S master and I2S slave modes through the configuration registers. The IP supports two channels with 32-bit audio resolution by default. The DAC/ ADC resolution can be configured as 8/16/24/32 bit wide.

The Arasan I2S Controller IP Core provides a 32-bit parallel processor bus as the application interface. The controller's I2S interface consists of one transmitter and one receiver. Each channel can be programmed as an I2S master or an I2S slave. The Bit Clock (BLCK) and Left and Right Clock (LRCK) provide synchronization to transmit and receive data. The I2S Controller IP supports 44.1KHz audio sampling rates. DAC/ADC resolution is configurable from 8-bit to 32-bit. The I2S Controller IP supports a 32-bit parallel bus interface. AHB, PCI or other custom specific buses can also be provided upon request.

10.2 Features

- Complies with Philips* I2S Specification
- Supports two I2S channels
- Simultaneous audio playback and recording
- Supports configurable 8/16/24/32 bit DAC/ADC resolution
- Supports 44.1KHz audio sampling frequencies
- 32-bit parallel processor bus
- Interrupt support for FIFO transfers
- Supports 256 sampling frequency operating modes
- Other custom buses available upon request

10.3 Architecture

The principle components of the Arasan Parallel I2S Controller are Parallel Bus Interface, Transmit FIFO, Receive FIFO, I2S Controller, Transmitter, and Receiver. The below figure depicts the block diagram of the I2S Controller.

10.3.1 Functional Block Diagram

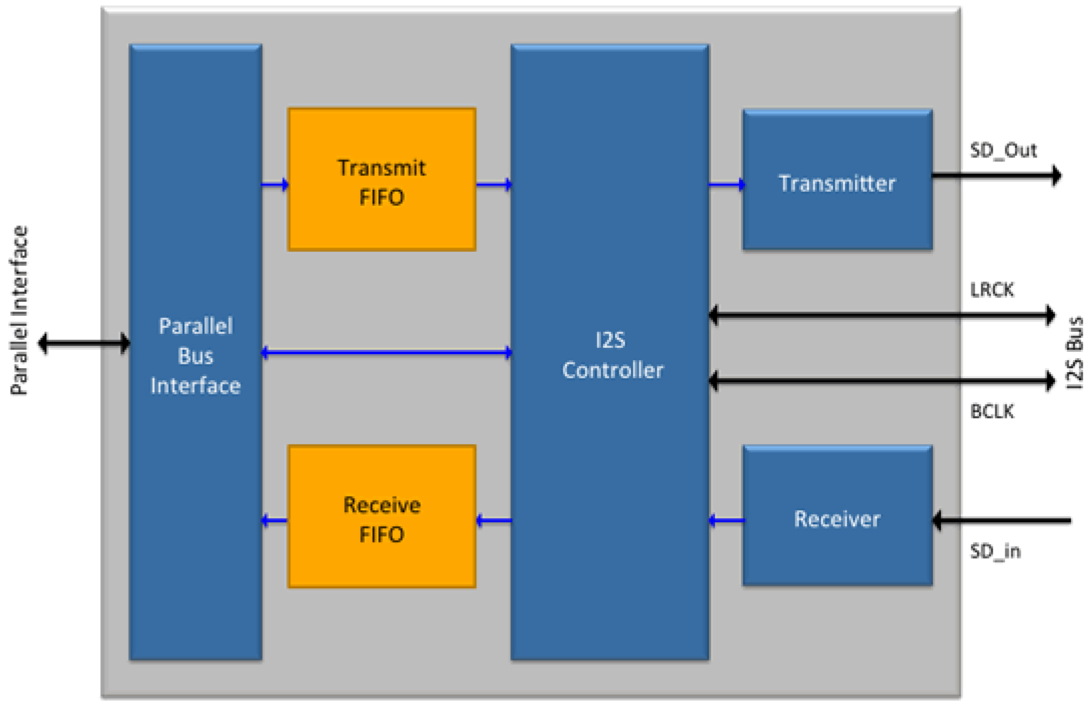

Figure 18: I2S Controller Core IP Block Diagram

10.3.2 Functional Block Diagram Decription

10.3.2.1 Parallel Bus Interface

The Arasan I2S controller uses a parallel interface to connect to any processor's parallel bus. The I2S Controller provides a 32-bit parallel processor bus interface. It consists of various control and status registers. The processor can access these registers through the parallel bus that provides two 32-bit wide data buses for simultaneous read and write operations. The I2S transmitter and receiver can be enabled or disabled through the enable registers in this block. The transmit ready (FIFO empty) and receive data ready (FIFO full) interrupts can also be enabled or disabled. The transmission or reception of data can be programmed through the data port register.

10.3.2.2 Controller

The I2S Controller controls the operation of the I2S transmitter and receiver. It directs data from the transmit FIFO to the transmitter and from the receiver to the receive FIFO. It also generates the bit clock (BCLK) and left-right clock (LRCK). These clocks are generated from a crystal oscillator source.

10.3.2.3 Transmit FIFO

The transmit FIFO and receive FIFO provide a means of data buffering. It stores data from the source until it is read by the sink. The FIFO's depth can be parameterized. To the transmit FIFO, the parallel bus is the source and transmitter is the sink. The receive FIFO stores data from the receiver until it is read by the parallel bus.

10.3.2.4 Receive FIFO

The I2S receiver stores data from SD_in input into the receive FIFO. It converts serial data from the SD_in input to parallel format and then stores it into the receive FIFO.

10.3.2.5 Transmitter

The I2S transmitter reads the 32-bit wide data from the transmit FIFO through the I2S controller. It shifts data serially out the SD_out line. Data shifting out the SD_out line is aligned to the LRCK clock output.
The transmitter interface can be programmed to operate as an I2S master or an I2S slave. There are two operating modes.

- Master Transmitter Mode - serial data is transmitted through SD_out while BCLK and LRCK are serial and word select clocks outputs respectively.
- Slave Transmitter Mode - serial data is transmitted via SD_out while BCLK and LRCK are serial and word select clocks inputs respectively.

10.3.2.6 Receiver

The receiver interface can be programmed to operate as an I2S master or an I2S slave. There are two operating modes.

- Master Receiver Mode - serial data is received via SD_in while BCLK and LRCK are serial and word select clocks outputs respectively.
- Slave Receiver Mode - serial data, serial clock and word select are received through SD_in, BCLK, and LRCK inputs respectively.

10.4 Signal Interfaces

The Arasan I2S Controller IP Core has a parallel interface and an I2S interface.

Table 1: Parallel Interface

Pin	Direction	Description
sys_clk	Input	System Clock
rst_n	Input	Active Low System Reset
cs_n	Input	Active Low Chip Select
addr[2:0]	Input	System Address Bus
data_in[31:0]	Input	Parallel Data Input
data_out[31:0]	Output	Parallel Data Output
nRD_WR	Input	Read/Write 0: Read 1: Write

Table 2: I2S Interface

Pin	Direction	Description
clk_48	Input	48 MHz Crystal Oscillator Input
bclk	Output	Audio bit clock generated when I2S controller is in master mode
lrck	Output	Word Select generated when I2S controller is in master mode.
bclk_en	Output	Audio bit clock asserted when I2S controller is in master mode.
lrck_en OUT	Output	Word Select asserted when I2S controller is in master mode.
SD_out	Output	Serial Output
SD_in I	Input	Serial Input

10.5 SoC Level Integration

10.5.1 Verification Environment

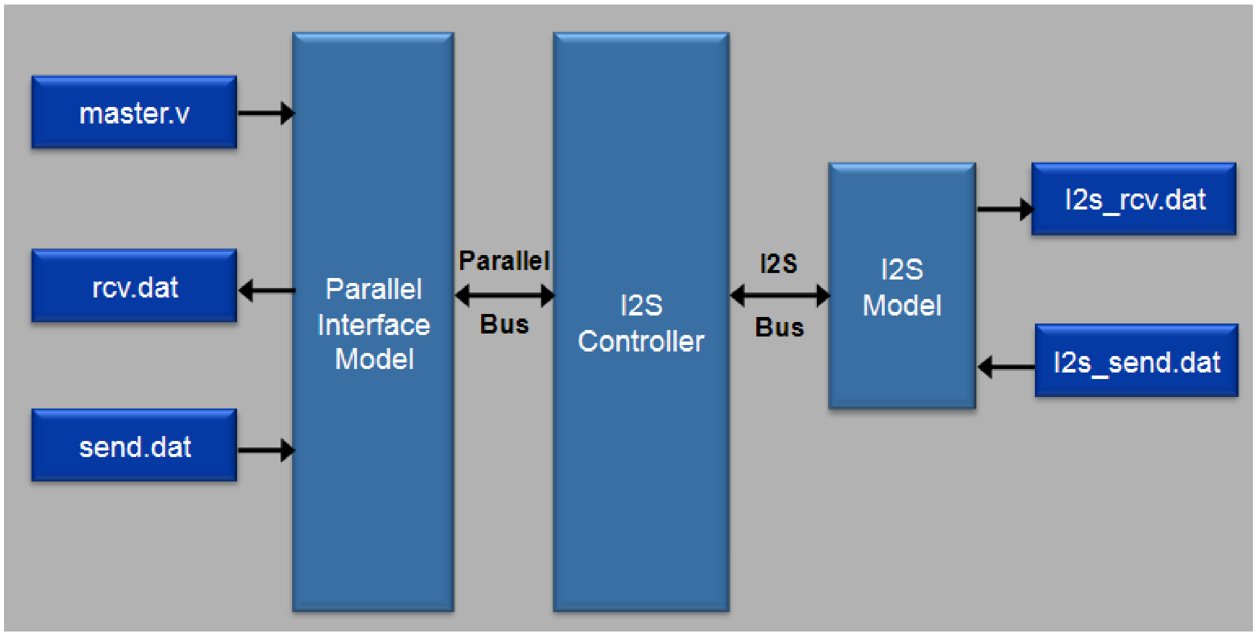

Figure 19: Verification Environment

10.5.2 IP Deliverables

The IP package consists of the following:

- RMM-compliant synthesizable RTL design in Verilog
- Easy-to-use test environment
- Synthesis scripts
- Technical documents

11 Services & Support

11.1 Global Support

Arasan Chip Systems provide global support to its IP customers. The technical support is not geographically bound to any specific site or location, and therefore our customers can easily get support for design teams that are distributed in several locations at no extra cost.

11.2 Arasan Support Team

Our technical support isprovided by the engineers who have designed the IP. That is a huge benefit for our customers, who can communicate directly with the engineers who have the deepest knowledge and domain expertise of the IP, and the standard to which it complies.

11.3 Professional Services & Customization

At Arasan Chip Systems we understand that no two Application Processors are the same. We realize that often the standard itself needs some tweaks and optimizations to fit your design better. Sometimes, the interface between the IP blocks and your design need some customization.Therefore, we provide professional services and customization to our IP customers. We do not sell our IP blocks as “black box” that cannot be touched. Please contact us for more details on our customization services.

11.4 The Arasan Porting Engine

Analog IP blocks, such as eMMC 5.1 HS400 PHY, are designed for a specific Fab and process technology. Arasan’s analog design team, utilizing its deep domain expertise and vast experience, is capable of porting the PHYs into any specific process technology required by the customer. That is “The Arasan Porting Engine”.

11.5 Pricing & Licensing

Arasan charges a one-time licensing fee, with no additional royalties.The licensing fee gives the right to use our IP for 1 project.Licensing fee for additional projects, using the same IP, is discounted.We also offer unlimited-use license.For any additional information regarding pricing and licensing – please contact our sales at: sales@arasan.com.