



Datasheet

SD 3.0/SDIO 3.0/eMMC v4.51 Total IP Solution

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

1	Introduction	1
1.1	About eMMC	1
1.2	Arasan's Contribution to JEDEC and SD Association.....	1
1.3	Arasan's Total IP Solution	2
2	SD 3.0/SDIO 3.0/eMMC 4.51 Host Controller IP	4
2.1	Overview.....	4
2.2	Features.....	4
2.3	Architecture.....	5
2.3.1	Functional Description.....	5
2.3.2	Functional Block Diagram	6
2.3.3	Functional Block Diagram Description.....	6
2.4	Pinouts.....	8
2.4.1	I/ O Description	8
2.5	SoC Level Integration.....	20
2.5.1	IP Deliverables for RTL Version.....	20
2.5.2	Verification Environment	20
3	SDIO 3.0 Device Controller IP	22
3.1	Overview.....	22
3.2	Features.....	22
3.3	Bus Topology	23
3.4	Architecture.....	26
3.4.1	Functional Block Diagram Description.....	26
3.5	Signals.....	28
3.6	SoC Level Integration.....	30
3.6.1	IP Deliverables	30
3.6.2	Verification Environment	30
4	SD/SDIO 3 Device Controller IP	31
4.1	Overview.....	31
4.2	Features.....	31
4.3	Bus Topology	32
4.3.1	SD-Combo Card Initialization Sequence	33
4.4	Architecture.....	39
4.4.1	Functional Block Diagram	39
4.4.2	Functional Block Diagram Description.....	40
4.5	Signals.....	41
4.5.1	Signal Interfaces	41

4.6	SoC Level Integration	44
4.6.1	IP Deliverables	44
5	eMMC 4.51 Device Controller IP	45
5.1	Overview.....	45
5.2	Features.....	45
5.3	Architecture.....	47
5.3.1	Functional Description.....	47
5.3.2	Functional Block Diagram	47
5.3.3	Functional Block Diagram Description.....	48
5.4	Signal Interfaces	49
5.5	SOC Level Integration	53
5.5.1	Verification Environment	53
5.5.2	Verification Deliverables	53
5.5.3	IP Deliverables	53
6	Hardware Validation Platform	54
6.1	Overview.....	54
6.2	Features.....	54
6.3	HVP Architecture	55
6.4	Deliverables	56
7	SD 3.0/SDIO 3.0/eMMC 4.51 Host Controller Software Stack	57
7.1	Overview.....	57
7.2	Features.....	57
7.3	Architecture.....	58
7.4	Deliverables	59
8	Services & Support	60
8.1	Global Support.....	60
8.2	Arasan Support Team	60
8.3	Professional Services & Customization.....	60
8.4	The Arasan Porting Engine	60
8.5	Pricing & Licensing.....	60

Tables

Table 1: AHB Bus Interface Signals	9
Table 2: OCP Bus Interface Signals	9
Table 3: AXI Bus Interface Signals	11
Table 4: SD3.0/SDIO3.0/eMMC4.51 Interface	14

Table 5: Power Control Signals (Used for SD/SDIO Mode only)	15
Table 6: Clock, Special Controls and Test Mode Signals	15
Table 7: Block RAM (SRAM) Interface Signals	16
Table 8: Core Configuration Signals	17
Table 9: SD Interface Signals	28
Table 10: AHB Slave Interface Signals	29
Table 11: AHB Master Interface Signals	29
Table 12: System Interface Signals	30
Table 13: SD Interface Signals	42
Table 14: AHB Slave Interface Signals	42
Table 15: AHB Master Interface Signals	43
Table 16: System Interface Signals	43
Table 17: eMMC Bus Interface Signals	49
Table 18: AHB Target Interface Signals:	50
Table 19: AHB Master Interface Signals	50
Table 20: 128x32 Dual-Port RAM1 Interface Signals	51
Table 21: 128x32 Dual-Port RAM2 Interface Signals	52
Table 22: 128x32 RAM Interface Signals	52

Figures

Figure 1: Arasan's Total IP Solution	3
Figure 2: eMMC5.1/SD3.0/SDIO3.0 Host Controller Functional Block Diagram	6
Figure 3: Verification Environment	21
Figure 4: SDIO Card Initialization Sequence for SD Mode	24
Figure 5: SDIO Card Initialization Sequence for SPI Mode	25
Figure 6: SDIO3.0 Device Controller Architecture	26
Figure 7: Verification Environment of SDIO Controller IP with AHB System Bus	30
Figure 8: SD Combo card initialization flow in SD mode	34
Figure 9: SD Combo card initialization flow in SD mode Continued	35
Figure 10: SD Combo card Initialization flow in SPI mode	37
Figure 11: SD Combo card Initialization flow in SPI mode Continued	38
Figure 12: SD_Combo_AHB Architecture	39
Figure 13: eMMC 4.51 Device Controller Functional Block Diagram	47
Figure 14: Verification Environment of eMMC 4.51 Device	53
Figure 15: SD/SDIO/eMMC HVP Architecture	55
Figure 16: SD 3.0/eMMC 4.51 Stack Architecture	59

1 Introduction

1.1 About eMMC

eMMC, short for "embedded Multi-Media Card", is an embedded non-volatile memory system, comprised of both flash memory and a flash memory controller integrated in the same industry-standard BGA package.

eMMC architecture, integrating the flash memory and controller in the same package, simplifies the application interface design and frees the host processor from low-level flash memory management. This benefits product developers by simplifying the non-volatile memory interface design and qualification process – resulting in reducing time-to-market, as well as future proofing against new flash memory technology advances.

The eMMC standard has been developed and published by JEDEC™ Solid State Technology Association (www.jedec.org), the global leader in the development of standards for the microelectronics industry. JEDEC has over 4,000 participants, representing nearly 300 companies, working together in 50 JEDEC committees.

The latest revision of the JEDEC standard is 5.0, released on Sep. 2013, and defines a maximum bandwidth of 400 MB/s over 8 data lanes. The combination of 200 MHz DDR clock rate, and 8 data lanes, requires the use of a hard PHY. Arasan has designed the analog PHY in 40nm, 28nm, and 16nm FinFET+ process technologies, and are all silicon proven. Latest eMMC revision of the JEDEC about to be released is eMMC5.1 (Item # JC-64.1-67.14).

1.2 Arasan's Contribution to JEDEC and SD Association

Arasan Chip Systems has been an executive member with the Multi Media Card Association (MMCA) since 2002. The MMCA was later merged with JEDEC in 2008 as the emphasis of MMCA shifted from removable storage to embedded storage for mobile devices. Arasan is currently a contributing member to JEDEC actively promoting both the eMMC and UFS standards. Arasan is also a contributing member to the SD Association since 2001.

Arasan is the leader of mobile storage, with 300 IP licensees since 2002 for SD, SDIO, NAND, eMMC and UFS. Our eMMC Host and Device IPs were licensed to both Application Processor companies like Intel, LG, Samsung and Huawei, as well as the majority of the Memory companies, and includes SK Hynix, among others.

Arasan's active involvement and contribution to the relevant standards bodies, lead to deep domain expertise, which in turn results in early availability of high quality IP for our customers.

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

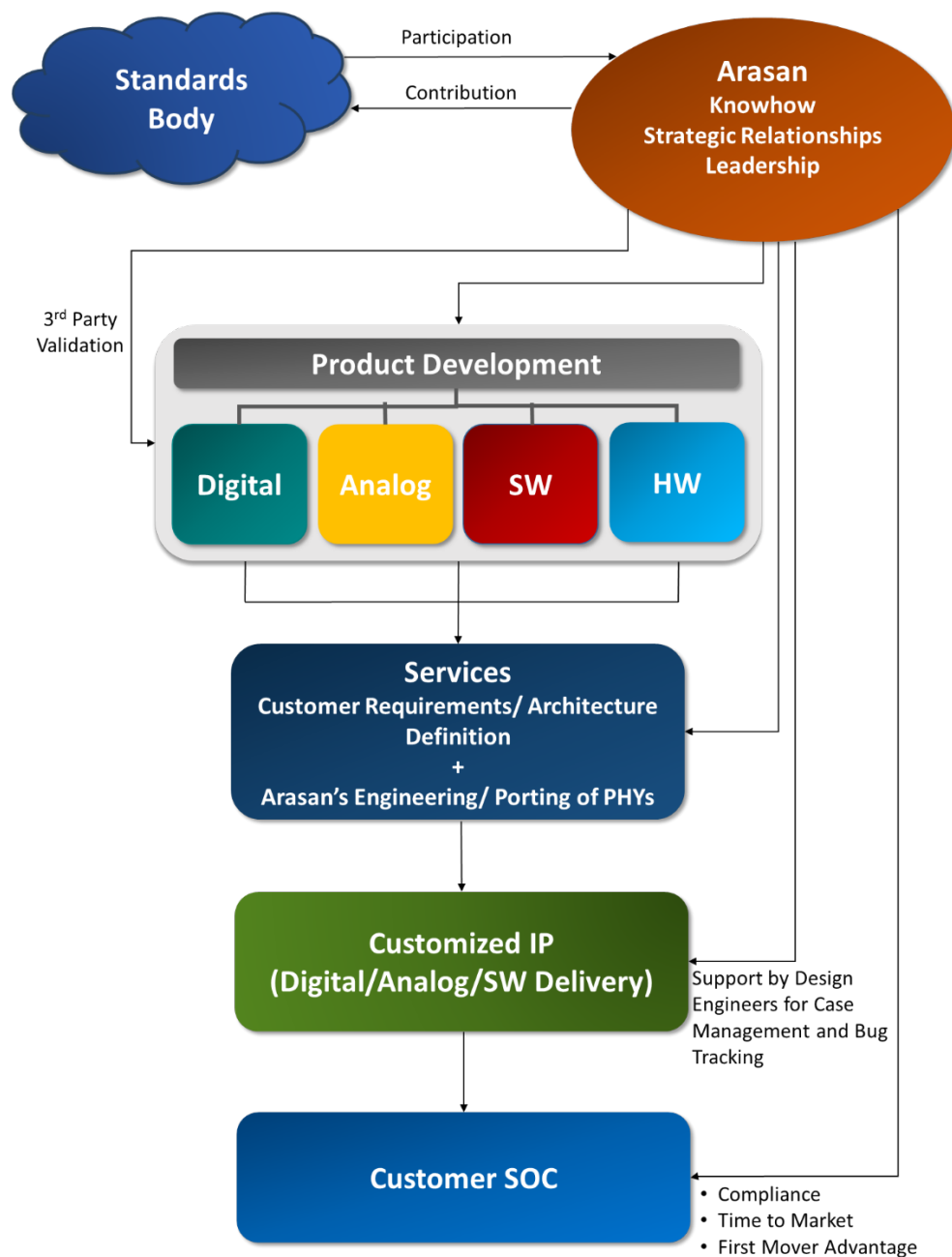


Figure 1: Arasan's Total IP Solution

2 SD 3.0/SDIO 3.0/eMMC 4.51 Host Controller IP

2.1 Overview

Arasan Chip Systems' SD 3.0/SDIO 3.0/eMMC 4.51 Host Controller IP is a highly integrated Intellectual Property (IP) solution that supports three key memory and I/O technologies:

1. SDA Secured Digital (SD) 3.0
2. SDA Secured Digital Input Output (SDIO) 3.0
3. JEDEC eMMC 4.51

This IP handles all the timing and interface protocol requirements to access these media as well as processing the commands in hardware.

The IP supports connection to a single slot and performs multi-block writes and erases that lower access overhead. In addition, a host can utilize this IP to boot directly from an attached eMMC device, thereby simplifying system initialization during power up. The host interface is based on a standard 32-bit Advanced High-Performance Bus (AHB) which is used to transfer data and configure the IP.

2.2 Features

- Compliant with the following specifications:
 - SDA Part A2 SD Host Controller Version 3.00
 - SDA Part 1 Physical Layer Specification Version 3.00
 - SDA Part E1 SDIO Specification Version 3.00
 - JEDEC eMMC Specification Version 4.51
 - AMBA, AHB Specification Version 2.00
 - AMBA, Advanced Extensible Interface (AXI) Specification Version 1.00 (Optional)
 - Open Core Protocol (OCP) Specification Version 2.2 (Optional)
- The core supports:
 - 32-bit and 64-bit system data bus
 - Interrupts and wake up functionality
 - Internal Clock divider for various card operational modes
 - One of the AHB, AXI or OCP System/Host bus
- The data is transferred using:
 - Programmed Input/Output (PIO) mode on the Host Bus Slave interface
 - Direct Memory Access (DMA) on the Host Bus Master interface*
 - Configurable FIFO size to support different block sizes.

Note: The Host Bus is AHB or AXI or OCP.

- UHS-I features (SD3.0/SDIO3.0) supports:
 - 1.8V voltages switch operation
 - Tuning for SDR104 mode and SDR50
 - Host clock rate variable between 0 and 208 MHz
 - Up to 832 Mbps data rate using 4 parallel data lines (SDR104 mode)
 - Transfers the data in 1-bit and 4-bit SD modes and SPI mode
 - Transfers the data in SDR104, DDR50, SDR50, SDR25, SDR12, DS and HS modes
 - Cyclic Redundancy Check (CRC): CRC7 for commands, CRC16 for data integrity
 - Variable-length data transfers
 - Performs Read Wait Control, Suspend/Resume operation with SDIO CARD
 - Designed to work with I/O cards, Read-only cards and Read/Write cards
 - Card Detection (Insertion/Removal)
- eMMC 4.51 features:
 - eMMC 4.51 security protocol commands
 - Primary and alternate boot modes
 - Packed commands, Data Tags, Discard and Sanitize features
 - 4KB block support
 - Tuning for HS200 mode
 - MMC Plus and MMC Mobile
 - Host clock rate variable between 0 and 200 MHz
 - Up to 1.6Gbps (HS200) data rate using 8 bit parallel data lines
 - Transfers the data in 1-bit, 4-bit and 8-bit modes
 - CRC7 for command and CRC16 for data integrity
 - Password protection of cards

2.3 Architecture

2.3.1 Functional Description

The Arasan SD 3.0/SDIO 3.0/ eMMC 4.51 Host Controller is a Host Controller with an AHB/AXI/OCP processor interface. This product conforms to SD Host Controller Standard Specification Version 3.00. It is also compliant to eMMC 4.51 Specification from JEDEC.

The SD3.0/SDIO3.0/eMMC4.51 Host Controller handles SDIO/SD Protocol at transmission level, packing data, adding CRC, Start/End bit, and checking for transaction format correctness. The Host Controller provides Programmed IO method and DMA data transfer method. In programmed IO method, the Host processor transfers data using the Buffer Data Port Register.

The SD3.0/SDIO3.0/eMMC4.51 Host Controller support for DMA can be determined by checking the DMA support in the Capabilities register. DMA allows a peripheral to read or write memory without the intervention from the CPU. This Host Controller's Host

Controller system address register points to the first data address, and data is then accessed sequentially from that address. It supports connection to a single slot and performs multi-block writes and erases the lower access.

2.3.2 Functional Block Diagram

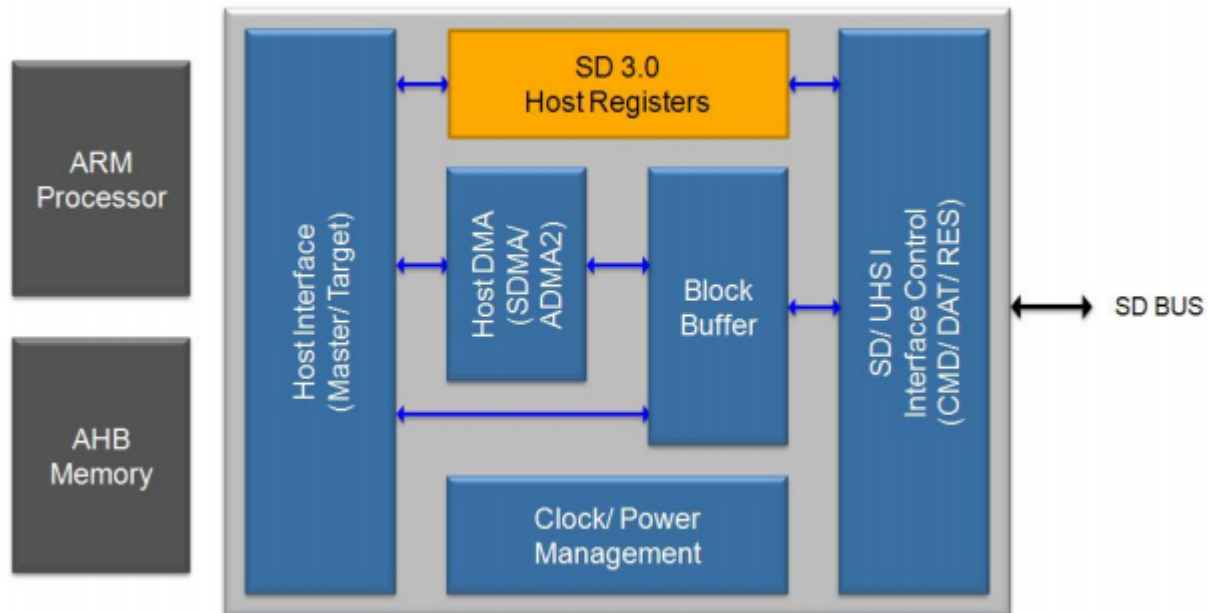


Figure 2: eMMC5.1/SD3.0/SDIO3.0 Host Controller Functional Block Diagram

2.3.3 Functional Block Diagram Description

2.3.3.1 Host Interface (Master/ Target)

The Master Bus Interface is used to access the DMA Controller (when using DMA) or Advanced Direct Memory Access (ADMA2 Modes). The DMA Controller module interfaces with the Host (AHB/AXI/OCP) Master Module to generate Transfers and on the other side it interfaces with the Block Buffer to store/fetch block data. The DMA Controller implements a separate DMA for Simple Direct Memory Access (SDMA) Operation and Separate DMA for the ADMA2 Operation. In addition, it implements Host Transaction Generator that generates controls for the Host Master Interface Module.

The DMA Controller uses the Master DMA interfaces to transfer data between the Host Controller and the System Memory and vice-versa and also to fetch the descriptors while operating in ADMA2 mode.

The Host Controller interfaces with the System bus using the AHB, AXI, or OCP Master and Slave Interface. The Slave Interface is used to access the Registers inside the Host controller. The Slave Interface supports only single transfer access (no Burst Support) and only one outstanding Read/Write transaction in case of AXI or OCP interface.

2.3.3.2 Host DMA

The PIO/DMA Controller module implements the SDMA and ADMA2 engines as defined in the SD Host Controller specification and maintains the block transfer counts for PIO operation. It interacts with the Registers Set and starts the DMA engine when a Command with Data Transfer is involved.

The DMA Controller interfaces with the Host (AHB/AXI/OCP) Master module to generate Transfers and on the other side it interfaces with the Block Buffer to store/fetch block data. It implements a separate DMA for SDMA operation and separate DMA for the ADMA2 operation. In addition implements Host Transaction Generator that generates controls for the Host Master interface module.

2.3.3.3 SD 3.0 Host Register

The Host Controller Register Set implements the Registers defined by the SD Host Controller Specification. The Registers are Byte/DWORD accessible from the Slave interface. The Host Controller Register Set also implements the Data Port Registers for the PIO Mode transfers.

The SD/SDIO Host Controller uses a Dual Port Block Buffer (read/write on both ports) that is used to store the Block Data during SD Transfers. The size of the Block Buffer is Configurable and has to be a minimum of 1 Block Size (Block Size is 512 Bytes in SD Memory and up to 2K Bytes in SDIO).

The Register Set provides the control signals and monitors the status signals from the blocks to set Interrupt Status Bits and eventually generate Interrupt signal to the Host Bus.

2.3.3.4 Block Buffer

The SD/SDIO Host Controller uses a Dual Port Block Buffer (Read/Write on both ports) or a Two Port (One Read/One Write) that is used to store the Block Data during SD Transfers. The size of the Block Buffer is Configurable and has to be a minimum of 1 Block Size (Block Size is 512 Bytes in SD Memory and up to 2K Bytes in SDIO).

To achieve maximum performance the Block buffer has to be sized to twice the maximum Block Size supported by Host Controller. The Block Buffer uses Circular Buffer Architecture. One side of the Block Buffer is interfaced with the DMA Controller and operates on the Host Clock. The other side of the Block Buffer interfaces with SD Interface Control Logic and operates on SD Clock. During a write transaction (data transferred from a Host Processor to SD3.0/SDIO3.0/ eMMC 4.51 card), the data is fetched from the Host System Memory and is stored in the Block Buffer. When a Block of data is available, the SD Control logic will transfer it onto the SD Interface.

The DMA Controller continues to fetch additional block of data when the Block Buffer has space. During a Read transaction (data transferred from SD3.0/SDIO3.0/ eMMC 4.51 card to Host Processor), the data from SD3.0/SDIO3.0/ eMMC 4.51 card will be written in to Block Buffer and at the end when the CRC of the Block is valid, the data is committed. When a Block of data is available, then the DMA Controller transfers this data to the Host System Memory. The SD Interface Control logic meanwhile receives the next Block of data provided there is space in the Block Buffer. If the

Host controller cannot accept any data from SD3.0/SDIO3.0/ eMMC 4.51 card, then it will issue Read Wait (if card supports Read Wait mechanism) to stop the data transfer from card or by stopping the clock.

Note: FIFO depth can be varied using parameter passed to the Core using the ‘dot parameter instantiation’. When the Block Buffer size is twice that of the Block Size, the Block Buffer behaves like a ping-pong buffer.

2.3.3.5 SD (UHS – I) Interface Control (CMD/ DAT/ RES)

The SD Interface Control block maps the internal signals to the External SD Interface and vice versa. Based on the Bus Width (1/4/8) the internal signals are driven out appropriately. In case of DS, the outputs are driven on the negative edge of the sd_clk.

The input from RxFlops module are latched on the rx_clk (looped back or tuned clock) and is output to the Receive Control Module for further processing.

2.3.3.6 Clock/ Power Management

The SD Interface Clock Generator module generates the SD Clock from the Reference Clock (xin_clk), based on the Controls programmed in the Clock Control Register. These include the Clock Divide Value, Clock Enable and so on. The outputs from this module are the SD_CLK and the CARD Clock. The SD_CLK is used by the eMMC/SD Interface Control Logic and the CARD Clock connected to the “CLK” Pin on the EMMC Interface. This module also generates system resets to various clock domains.

2.4 Pinouts

2.4.1 I/ O Description

The Arasan SD3.0/SDIO3.0/ eMMC 4.51 Host Controller has the following interface groups.

- System (AHB/AXI/OCP) Bus Interface Signals
- SD3.0/SDIO3.0/ eMMC 4.51 Interface that forms the main card interface
- Power Control Signals
- Clock, Special Controls and Test Mode Signals
- Block RAM, Static Random Access Memory (SRAM) Interface Signals
- Core Configuration Signals

Table 1: AHB Bus Interface Signals

Pin	Direction	Description
ahb_clk	In	AHB System Clock
ahb_reset_n	In	AHB System Reset (Active Low)
ahbmaster_hbusreq	Out	AHB Bus request
ahbmaster_hgrant	In	AHB Bus Grant
ahbmaster_haddr[AW-1:0]	Out	DWord Address
ahbmaster_hwdata[DW-1:0]	Out	AHB master write data
ahbmaster_hrdata[DW-1:0]	In	AHB master read data
ahbmaster_hwrite	Out	Write / Read Direction Indication
ahbmaster_hsize[2:0]	Out	Size (byte, half word or word)
ahbmaster_hburst[2:0]	Out	Burst Size
ahbmaster_hready	In	Ready signal
ahbmaster_htrans[1:0]	Out	Transfer type
ahbmaster_hresp[1:0]	In	Transfer response
ahb_intr	Out	Interrupt to the ARM
ahb_wkup	Out	Wakeup Indication to ARM
ahbtarget_hsel	In	Slave Select
ahbtarget_haddr[15:0]	In	DWord Address (256 bytes)
ahbtarget_hwdata[31:0]	In	Write Data
ahbtarget_hrdata[31:0]	Out	Read Data
ahbtarget_hwrite	In	Write / Read Direction Indication
ahbtarget_hsize[2:0]	In	Size (Byte, Half Word or Word)
ahbtarget_htrans[1:0]	In	Transfer Type
ahbtarget_hready_in	In	Slave Ready Input
ahbtarget_hready	Out	Slave Ready
ahbtarget_hresp[1:0]	Out	Transfer Response

Note: Target interface doesn't support BURST transaction.

Table 2: OCP Bus Interface Signals

Pin	Direction	Description
clk_ocp	In	OCP System Clock.
OCPMaster_MAddr[AW-1:0]	Out	OCP Master read/write address.
OCPMaster_MCmd[2:0]	Out	Indicates the type of transaction that the OCP Master has initiated
OCPMaster_MData[DW-1:0]	Out	Write data from OCP Master to the slave
OCPMaster_MDataValid	Out	Is the qualifier for OCPMaster_MData
	In	Indicates that the OCP Slave has accepted the command
OCPMaster_SData[DW-1:0]	In	Read data from OCP Slave
OCPMaster_SDataAccept	In	Asserted by OCP slave to indicate that the current Master write data is accepted

OCPMaster_SResp[1:0]	In	Response signal for Master write transfers
OCPMaster_MByteEn[3:0]	Out	Byte enable from the Master for write/read transactions
OCPMaster_MBurstLength[4:0]	Out	Indicates the burst length of the transaction
OCPMaster_MBurstPrecise	Out	Indicates that the given burst length is precise
OCPMaster_MBurstSeq[2:0]	Out	Indicates the type of burst
OCPMaster_MBurstSingleReq	Out	Indicates the number of requests associated with the burst
OCPMaster_MDataLast	Out	Last Data of the burst
OCPMaster_MReqLast	Out	Last request in a burst
OCPMaster_SRespLast	In	Last response in a burst
OCPMaster_MDataByteEn[3:0]	Out	Write Byte enables the OCP slave during Data handshake phase
OCPMaster_max_burst_size_config	In	Configurable burst length
OCPSlave_MCmd[2:0]	In	Type of transaction from the Master
OCPSlave_MAddr[31:0]	In	Transfer address from Master
OCPSlave_MData[31:0]	In	Write data from OCP Master
OCPSlave_SCmdAccept	Out	Acceptance signal to the external OCP Master for the request phase
OCPSlave_SResp[1:0]	Out	Response signal from OCP Slave
OCPSlave_SData[31:0]	Out	Read data from OCP Slave
OCPSlave_MByteEn[3:0]	In	Byte Enable from OCP Master
OCPSlave_MReset_n	In	Reset signaling from OCP Master
OCPSlave_MRespAccept	In	Master accepts response

Table 3: AXI Bus Interface Signals

Pin	Direction	Description
aximst_arid[3:0]	Out	Read address ID. This signal is the identification tag for the read address group of signals.
aximst_araddr[AW-1:0]	Out	Read address. The read address bus gives the initial address of a read burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst
aximst_arlen[3:0]	Out	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
aximst_arsize[2:0]	Out	Burst size. This signal indicates the size of each transfer in the burst.
aximst_arburst[1:0]	Out	Burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
aximst_arvalid	Out	Read address valid. This signal indicates, when HIGH, that the read address and control information is valid and will remain stable until the address acknowledge signal, ARREADY, is high. 1 = address and control information valid 0 = address and control information not valid.
aximst_rid[3:0]	In	Read ID tag. This signal is the ID tag of the read data group of signals. The RID value is generated by the slave and must match the ARID value of the read transaction to which it is responding.
aximst_rdata[DW-1:0]	In	Read data.
aximst_rresp[1:0]	In	Read response. This signal indicates the status of the read transfer. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.
aximst_rlast	In	Read last. This signal indicates the last transfer in a read burst.
aximst_rvalid	In	Read valid. This signal indicates that the required read data is available and the read transfer can complete: 1 = read data available 0 = read data not available
aximst_rready	Out	Read ready. This signal indicates that the Master can accept the read data and response information: 1 = Master ready 0 = Master not ready.
axislv_awid[7:0]	In	Write address ID. This signal is the identification tag for the write address group of signals.

Pin	Direction	Description
axislv_awaddr[AW-1:0]	In	Write address. The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst.
axislv_awlen[3:0]	In	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
axislv_awsz[2:0]	In	Burst size. This signal indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update.
axislv_awburst[1:0]	In	Burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
axislv_awvalid	In	Write address valid. This signal indicates that valid write address and control information are available: 1 = address and control information available 0 = address and control information not available The address and control information remain stable until the address acknowledge signal, AWREADY, goes HIGH.
axislv_awready	Out	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready 0 = slave not ready.
axislv_wid[7:0]	In	Write ID tag. This signal is the ID tag of the write data transfer. The WID value must match the AWID value of the write transaction.
axislv_wdata[DW-1:0]	In	Write data.
axislv_wstrb[DW/8-1:0]	In	Write strobes. This signal indicates which byte lanes to update in memory. There is one write strobe for each eight bits of the write data bus. Therefore, WSTRB[n] corresponds to WDATA[(8 * n) + 7:(8 * n)].
axislv_wlast	In	Write last. This signal indicates the last transfer in a write burst.
axislv_wvalid	In	Write valid. This signal indicates that valid write data and strobes are available: 1 = write data and strobes available 0 = write data and strobes not available.
axislv_wready	Out	Write ready. This signal indicates that the slave can accept the write data: 1 = slave ready

Pin	Direction	Description
		0 = slave not ready.
axislv_bid[7:0]	Out	Response ID. The identification tag of the write response. The BID value must match the AWID value of the write transaction to which the slave is responding.
axislv_bresp[1:0]	Out	Write response. This signal indicates the status of the write transaction. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.
axislv_bvalid	Out	Write response valid. This signal indicates that a valid write response is available: 1 = write response available 0 = write response not available.
axislv_bready	In	Response ready. This signal indicates that the Master can accept the response information. 1 = Master ready 0 = Master not ready
axislv_arid[7:0]	In	Read address ID. This signal is the identification tag for the read address group of signals.
axislv_araddr[31:0]	In	Read address. The read address bus gives the initial address of a read burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst.
axislv_arlen[3:0]	In	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
axislv_arsize[2:0]	In	Burst size. This signal indicates the size of each transfer in the burst.
axislv_arburst[1:0]	In	Burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
axislv_arvalid	In	Read address valid. This signal indicates, when HIGH, that the read address and control information is valid and will remain stable until the address acknowledge signal, ARREADY, is high. 1 = address and control information valid 0 = address and control information not valid.
axislv_arready	Out	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready 0 = slave not ready.
axislv_rid[7:0]	Out	Read ID tag. This signal is the ID tag of the read data

Pin	Direction	Description
		group of signals. The RID value is generated by the slave and must match the ARID value of the read transaction to which it is responding.
axislv_rdata[31:0]	Out	Read data. The read data bus can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide.
axislv_rresp[1:0]	Out	Read response. This signal indicates the status of the read transfer. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.
axislv_rlast	Out	Read last. This signal indicates the last transfer in a read burst.
axislv_rvalid	Out	Read valid. This signal indicates that the required read data is available and the read transfer can complete: 1 = read data available 0 = read data not available.
axislv_rready	In	Read ready. This signal indicates that the Master can accept the read data and response information: 1= Master ready 0 = Master not ready.
int_to_arm	Out	Interrupt to the ARM
cfg_mstid	In	Programmable ID for Master interface

Table 4: SD3.0/SDIO3.0/eMMC4.51 Interface

Pin	Direction	Description
sdif_cd_n	In	Active Low. Card Detection for single Slot (optional. Used for SD/SDIO interface only)
sdif_wp	In	Active High. SD Card Write Protect (optional. Used for SD/SDIO Interface only)
sdif_clkout	Out	Clock to Card (CLK)
rxclk_in	In	Clock looped back from PAD
sdif_cmdin	In	Command Input
sdif_cmdout	Out	Command Output
sdif_cmdena	Out	Command Output Enable
sdif_dat0in	In	Data0 Input
sdif_dat0out	Out	Data0 Output
sdif_dat0en	Out	Data0 Output Enable
sdif_dat1in	In	Data1 Input or Interrupt (for SDIO)
sdif_dat1out	Out	Data1 Output
sdif_dat1en	Out	Data1 Output Enable
sdif_dat2in	In	Data2 Output or Read Wait (optional)
sdif_dat2out	Out	Data2 Output
sdif_dat2en	Out	Data2 Output Enable
sdif_dat3in	In	Data3 Input

Pin	Direction	Description
sdif_dat3out	Out	Data3 Output
sdif_dat3en	Out	Data3 Output Enable
sdif_dat4in	In	Data4 Input
sdif_dat4out	Out	Data4 Output
sdif_dat4en	Out	Data4 Output Enable
sdif_dat5in	In	Data5 Input
sdif_dat5out	Out	Data5 Output
sdif_dat5en	Out	Data5 Output Enable
sdif_dat6in	In	Data6 Input
sdif_dat6out	Out	Data6 Output
sdif_dat6en	Out	Data6 Output Enable
sdif_dat7in	IN	Data7 Input
sdif_dat7out	Out	Data7 Output
sdif_dat7en	Out	Data7 Output Enable

Note: CMD/DATA output enables are active high signals

Table 5: Power Control Signals (Used for SD/SDIO Mode only)

Pin	Direction	Description
sdhc_ledcontrol	Out	LED ON: To Caution the user not to remove the card while the SD card is being accessed.
sdhc_sdbuspower	Out	Control SD Card Power Supply.
sdhc_sdbusvoltage[2:0]	Out	SD Bus voltage select.
sdhc_1p8vsigenable	Out	1.8V Signaling Enable
sdhc_driverstrength[1:0]	Out	Driver Strength Select 00b Driver Type B is Selected 01b Driver Type A is Selected 10b Driver Type C is Selected 11b Driver Type D is Selected

Table 6: Clock, Special Controls and Test Mode Signals

Pin	Direction	Description
xin_clk	In	This clock input is used to generate eMMC/SD Clock. For maximum efficiency this should be around 200 MHz for eMMC or 208MHz (for SD3.0).
corectrl_itapdlyena	In	Used to enable selective Tap delay line on the Looped back eMMC/SD Clock (rxclk_in). This signal along with the corectrl_itapdlysel[4:0] selects the amount of delay to be inserted on the line. When Tuning is enabled (for HS200/SDR104), this signal is ignored and internal controls are used instead.

Pin	Direction	Description
		This should not be asserted when operating in DS mode.
corectrl_itapdlysel[4:0]	In	Selects one of the 32 Taps on the rxclk_in line. This is effective only when corectrl_itapdlyena is asserted and Tuning is not enabled.
corectrl_itapchgwin	In	This is used to gate the output of the Tap Delay lines so as to avoid glitches being propagated into the Core. This signal should be asserted few clocks before the corectrl_itapdlysel changes and should be asserted for few clocks after.
corectrl_otapdlyena	In	Used to enable the selective Tap delay on the card_clk so as to generate the delayed card_clk. This is used to latch the CMD/DAT outputs to generate delay on them w.r.t CLK going out. This signal along with corectrl_otapdlysel[3:0] selects the amount of delay to be inserted on the Clock line. This signal should not be asserted when operating in DS mode
corectrl_otapdlysel[3:0]	In	Selects one of the 16 Taps on the sdclock_clk. This is effective only when corectrl_otapdlyena is asserted.
test_mode	In	Test mode signal is used for DFT purpose. Muxes in the AXI_reset_n signal for all internally generated resets. (Active High)
scan_mode	In	Scan Mode signal for selecting Scan Clocks for internally generated clocks
scan_clk1	In	Scan Clock#1 used to mux in for the internally generated sd_clk
scan_clk2	In	Scan Clock#2 used to mux in for the final rxclk_in (after the tap delay etc)

Table 7: Block RAM (SRAM) Interface Signals

Pin	Direction	Description
sram_clka	Out	Clock for PORT A
sram_addra [N-1:0]	Out	Address bus for PORT A. The width of the Address bus is based on the size of the SRAM (SDHC_BUFFER_SIZE)
sram_writea	Out	Write Enable for PORT A
sram_reada	Out	Read Enable for PORT A
sram_wrdata [DW-1:0]	Out	Write Data for PORT A.
sram_rddataa [DW-1:0]	In	Read Data from SRAM on PORT A. N is based on the SDHC_MSTAXI_DW parameter
sram_clkb	Out	Clock for PORT B
sram_addrb [N-1:0]	Out	Address bus for PORT B.

		The width of the Address bus is based on the size of the SRAM (SDHC_BUFFER_SIZE)
sram_writeb	Out	Write Enable for PORT B
sram_readb	Out	Read Enable for PORT B
sram_wratab [DW-1:0]	Out	Write Data for PORT B.
sram_rddatab [DW-1:0]	In	Read Data from SRAM on PORT B.

Table 8: Core Configuration Signals

Pin	Direction	Description
corecfg_tuningcount[5:0]	In	Configures the Number of Taps (Phases) of the rxclk_in that is supported. The Tuning State machine uses this information to select one of the Taps (Phases) of the rxclk_in during the Tuning Procedure.
corecfg_timeoutclkfreq[5:0]	In	Timeout Clock Frequency Suggested Value is 1. (KHz or MHz). Internally the 1msec /1usecTimer is used for Timeout Detection. The 1msec Timer is generated from the xin_clk.
corecfg_timeoutclkunit	In	Timeout Clock Unit Suggested value is 1'b1 to Select MHz Clock.
corecfg_baseclkfreq[7:0]	In	Base Clock Frequency for SD Clock. This is the frequency of the xin_clk.
corecfg_maxblklength[1:0]	In	Max Block Length Maximum Block Length supported by the Core/Device 00: 512 (Bytes) 01: 1024 10: 2048 11: Reserved
corecfg_8bitsupport	In	8-bit Support for Embedded Device Suggested Value is 1'b1 (The Core supports 8-bit Interface). Optionally can be set to 1'b0 if the Application supports only 4-bit SD Interface.
corecfg_adma2support	In	ADMA2 Support Suggested Value is 1'b1 (The ADMA2 is supported by Core). Optionally can be set to 1'b0 if the application doesn't want to support ADMA2 Mode
corecfg_highspeedsupport	In	High Speed Support Suggested Value is 1'b1 (The High Speed mode is supported by Core).
corecfg_sdmasupport	In	SDMA Support Suggested Value is 1'b1 (The SDMA is supported by Core).

Pin	Direction	Description
		Optionally can be set to 1'b0 if the application doesn't want to support SDMA Mode
corecfg_suspressupport	In	Suspend/Resume Support Suggested Value is 1'b1 (The Suspend/Resume is supported by Core). Optionally can be set to 1'b0 if the application doesn't want to support Suspend/Resume Mode
corecfg_3p3voltsupport	In	3.3V Support Suggested Value is 1'b1 as the 3.3 V is the default voltage on the SD Interface.
corecfg_3p0voltsupport	In	3.0V Support Should be set based on whether 3.0V is supported on the SD Interface.
corecfg_1p8voltsupport	In	1.8V Support Suggested Value is 1'b1 (The 1.8 Volt Switching is supported by Core). Optionally can be set to 1'b0 if the application doesn't want 1.8V switching (SD3.0)
corecfg_64bitsupport	In	64-bit System Bus Support This should be set based on the System Address Bus. When set to 1'b0 the Core supports only 32-bit System Bus. When set to 1'b1 the Core supports 64-bit System Address.
corecfg_asyncintrsupport	In	Asynchronous Interrupt Support Suggested Value is 1'b1 (The Core supports monitoring of Asynchronous Interrupt)
corecfg_slottype[1:0]	In	Slot Type Should be set based on the final product usage 00 - Removable SCard Slot 01 - Embedded Slot for One Device 10 - Shared Bus Slot 11 - Reserved
corecfg_sdr50support	In	SDR50 Support Suggested Value is 1'b1 (The Core supports SDR50 mode of operation) Optionally can be set to 1'b0 if the application doesn't want to support SDR50
corecfg_sdr104support	In	SDR104 Support Suggested Value is 1'b1 (The Core supports SDR104 mode of operation) Optionally can be set to 1'b0 if the application doesn't want to support SDR104
corecfg_ddr50support	In	DDR50 Support Suggested Value is 1'b1 (The Core supports DDR50 mode of operation) Optionally can be set to 1'b0 if the application doesn't want to support DDR50

Pin	Direction	Description
corecfg_hs400support	In	HS400 Support Suggested Value is 1'b1 (The Core supports HS400 Mode). This applies only to eMMC5.0 mode. This should be set to 1'b0 for SD3.0 mode Optionally can be set to 1'b0 if the application doesn't want to support HS400
corecfg_adriversupport	In	Driver Type A Support This bit should be set based on whether Driver Type A for 1.8 Signaling is supported or not.
corecfg_cdriversupport	In	Driver Type C Support This bit should be set based on whether Driver Type C for 1.8 Signaling is supported or not.
corecfg_ddriversupport	In	Driver Type D Support This bit should be set based on whether Driver Type D for 1.8 Signaling is supported or not.
corecfg_retuningtimercnt[3:0]	In	Timer Count for Re-Tuning This is the Timer Count for Re-Tuning Timer for Re-Tuning Mode 1 to 3. Setting to 4'b0 disables Re-Tuning Timer.
corecfg_tuningforsdr50	In	Use Tuning for SDR50 This bit should be set if the Application wants Tuning be used for SDR50 Modes. The Core operates with or without tuning for SDR50 mode as long as the Clock can be manually tuned using tap delay.
corecfg_retuningmodes[1:0]	In	Re-Tuning Modes Should be set to 2'b00 as the Core supports only the Mode0 Retuning.
corecfg_spisupport	In	SPI Mode Support Suggested Value is 1'b1 (The Core supports SPI mode of operation) Optionally can be set to 1'b0 if the application doesn't want to support SPI Mode
corecfg_spiblkmode	In	SPI Block Mode Reserved and should be set to 1'b0
corecfg_type4support	In	Driver Type 4 Support This bit should be set to 1'b1 if the Host Controller supports Type4 Drive Strength (eMMC5.0), otherwise it should be set to 0.
corecfg_initpresetval[12:0]	In	Preset Value for Initialization.
corecfg_dsppresetval[12:0]	In	Preset Value for Default Speed
corecfg_hspresetval[12:0]	In	Preset Value for High Speed
corecfg_sdr12presetval[12:0]	In	Preset Value for SDR12
corecfg_sdr25presetval[12:0]	In	Preset Value for SDR25
corecfg_sdr50presetval[12:0]	In	Preset Value for SDR50

Pin	Direction	Description
corecfg_sdr104presetval[12:0]	In	Preset Value for SDR104
corecfg_ddr50presetval[12:0]	In	Preset Value for DDR50
corecfg_hs400presetval[12:0]	In	Preset Value for HS400
corecfg_maxcurrent1p8v[7:0]	In	Maximum Current for 1.8V
corecfg_maxcurrent3p0v[7:0]	In	Maximum Current for 3.0V
corecfg_maxcurrent3p3v[7:0]	In	Maximum Current for 3.3V
corecfg_asyncwakeupena	In	<p>Determines the Wakeup Signal Generation Mode.</p> <p>0: Synchronous Wakeup Mode: The xin_clk has to be running for this mode. The Card Insertion/Removal/Interrupt events are detected synchronously on the xin_clk and the Wakeup Event is generated. The Assertion and deassertion of the wakeup Event signal synchronous to xin_clk.</p> <p>1: Asynchronous Wakeup Mode: The xin_clk and the host_clk can be stopped in this mode and the Wake up Event is asynchronously generated based on the Card Insertion/Removal/Interrupt Events. The Assertion and de-assertion of the wakeup Event signal is asynchronous.</p>

2.5 SoC Level Integration

2.5.1 IP Deliverables for RTL Version

- Verilog HDL of the IP Core
- Synthesis scripts
- Test environment and test scripts
- User guide

2.5.2 Verification Environment

This section provides information about the architecture of the SD Host Controller Verification Environment.

The SD Host Controller Design Under Test (DUT) is written in synthesizable Verilog. On the processor side it interfaces with AXI/AHB/OCP Master BFM and AXI/AHB/OCP Slave BFM. On the device side it interfaces with a user selectable device BFM (SD, SDIO, eMMC). The Host Controller Capability is selected by connecting the capability pins to power and ground. The device side interface is 8 bits wide bi-directional data lane supporting DDR (Double Data Rate). The width of the bus is user selectable (1, 4 or 8). The data can be sent or received either by processor input and output or by DMA. The data flow can be aborted by the processor side sending an abort command. The data transfer goes through a ping-pong buffer which achieves back to back frames transfer.

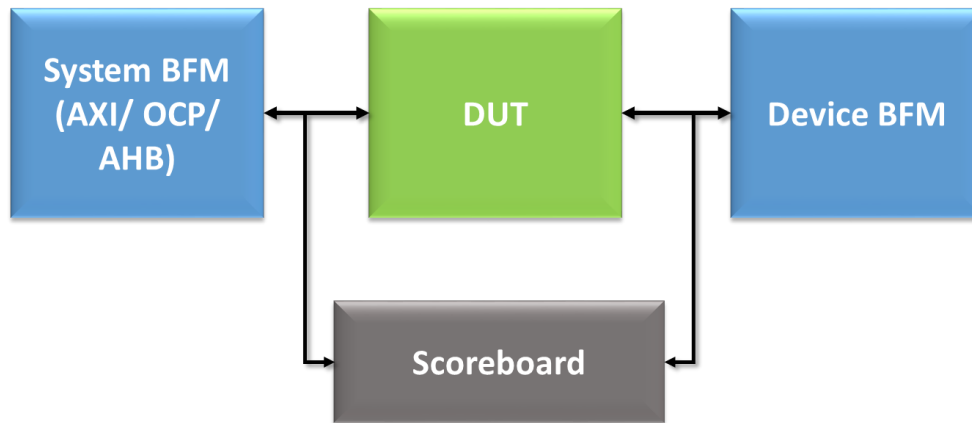


Figure 3: Verification Environment

3 SDIO 3.0 Device Controller IP

3.1 Overview

The Arasan SDIO 3.0 Device Controller is a full / high speed card controller suitable for I/O Card applications like WLAN, Bluetooth with low power consumption for mobile devices. The controller supports SPI, 1-bit SD, 4-bit SD and 8-bit bus modes for Embedded SDIO cards. The Arasan SDIO Device Controller has an AHB interface, which allows the ARM Processor to configure the operational registers residing inside the AHB Slave core.

The SDIO 3.0 Device IP core is used to implement SDIO cards that are connected to a SD host or card reader over a standard SD bus. The flexible architecture of the SD IO Device IP core is targeted to develop a range of portable, low-power cards such as the WiFi, GPS and LTE.

The SDIO 3.0 Device IP core is fully compliant with the SD Specification Part E1 SDIO 3.0. It supports SPI, SD1 and SD4 bit transfer modes and multiple functions per card. High-speed and full-speed SD data transfers are also supported. All SDIO 3.0 features are supported including the UHS-I, SDHC, mini SDIO, embedded SDIO ATA standard function interface code and operating voltages 2.7-3.6V or 1.7-1.95V. In applications with an AHB interface, the SDIO Device controller IP is controlled by an ARM processor. The SDIO Device controller includes a bidirectional FIFO that is expandable from 4 x 32-bit to any size required. The core supports asynchronous interrupts to the SD Host for improved performance. It supports suspend/resume operation for improved performance.

The controller integrates a scatter gather DMA engine automating data transfers between the SDIO card and system memory. The SDIO Device core is available with many system bus interfaces including AHB, AXI, OCP and custom buses. The wide selection of system bus interfaces enables the core to integrate effectively SoC designs today.

3.2 Features

- Compliant with SD Specification Part E1 SDIO Specification 3.0
- Supports Asynchronous Interrupt to Host controller
- Enhanced power management using new Power State Control function
- Supports Read Wait Control, Suspend/ Resume operations for superior card performance
- High-performance UHS-I (SDR104 up to 104MB/s)
- Multiple I/O functions and one memory supported
- Host clock rate from 0 to 208 MHz
- Supports SPI, 1-bit and 4-bit SD modes
- Optional 8-bit mode for embedded SDIO
- Supports all SDIO form factors including standard, mini and micro SDIO card
- Embedded SDIO ATA interface code
- Bus Master with Scatter Gather DMA
- Dual operating voltage range 2.7V – 3.6V and 1.7V – 1.95V

- Up to 104 MB/s read/write with 4-bit data lines in SD4 mode
- Cyclic Redundancy Check (CRC7) (command), CRC16 (data) integrity
- Supports direct R/W (IO52) and extended R/W (IO53)
- Programmable through AMBA 2.0 AHB bus

3.3 Bus Topology

The SD Card defines two alternative communication protocols SD and SPI.

1. SD

- SD 1-bit
- SD 4-bit

2. SPI

Applications can choose either one of modes. UHS-I cards does not support SPI mode and SD 1-bit mode. Mode selection is transparent to the Host. After reset command the Card automatically moves to default mode and will expect all further communication to be in the same mode.

Therefore, applications which uses only one communication mode do not have to be aware of the other.

SD

SD bus allows dynamic configuration of the number of data lines. After power up by default, the SD card will use one DAT0 for data transfer. After initialization the Host can change the bus width (number of active data lines). This feature allows easy trade-off between HW cost and system performance.

8-bit bus mode can be supported only for an embedded device. Achieving higher performance in the lower frequency is the objective of this mode. In SD mode DAT0 is used for data transfer and busy signalling, DAT1 is used for data transfer and Interrupt signaling, DAT2 is used for data transfer and read_wait, DAT3 is used for data transfer.

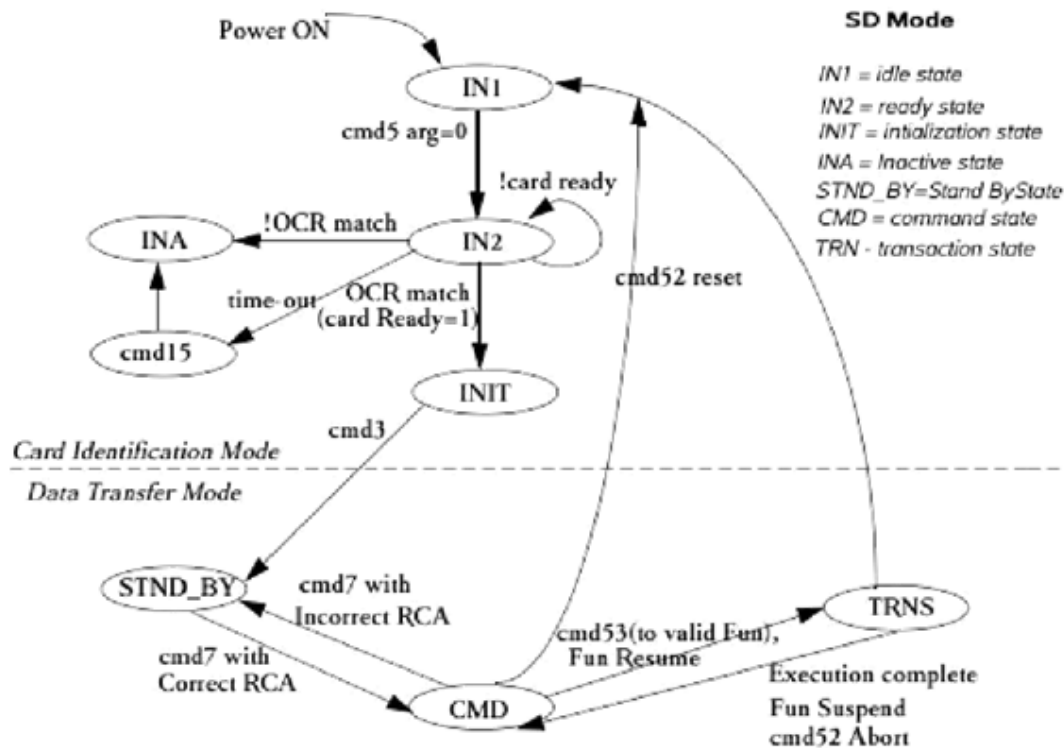


Figure 4: SDIO Card Initialization Sequence for SD Mode

SPI

The SPI mode is selected during the first reset command after power up and cannot be changed as long as the part is powered on. From the application point of view, the advantage of the SPI mode is the capability of using an off-the-shelf host, hence reducing the design-in effort to minimum. The disadvantage is the loss of performance, relatively to the SD mode which enables the wide bus option. The SPI interface uses the 8 out of the SDIO 9 signals (DAT 2 is not used, DAT1 for interrupt signaling, DAT3 for CS signaling) of the SD bus.

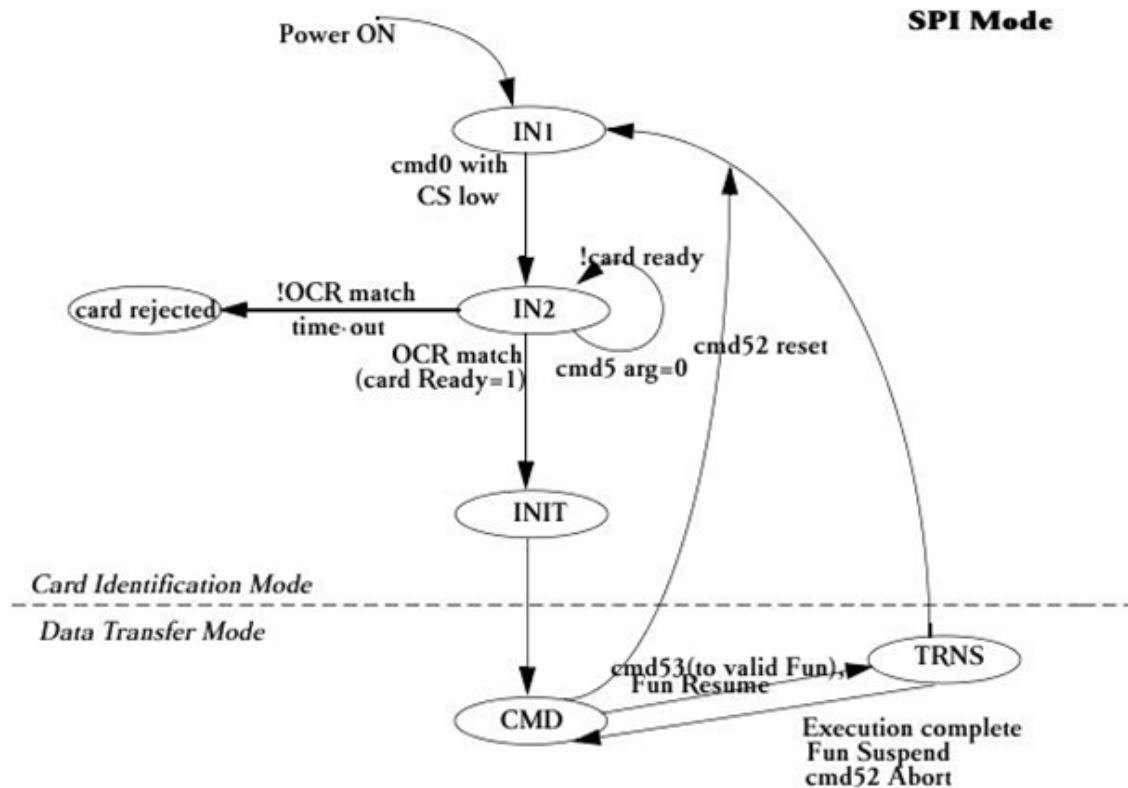


Figure 5: SDIO Card Initialization Sequence for SPI Mode

Note:

- To access the card being already in inactive state, a hard reset must be done by switching the power supply off and on.
- Card ready time-out: Time-out for Initialization process is 1 second

3.4 Architecture

This section contains the architecture of SDIO Device Controller and the description of internal blocks in detail.

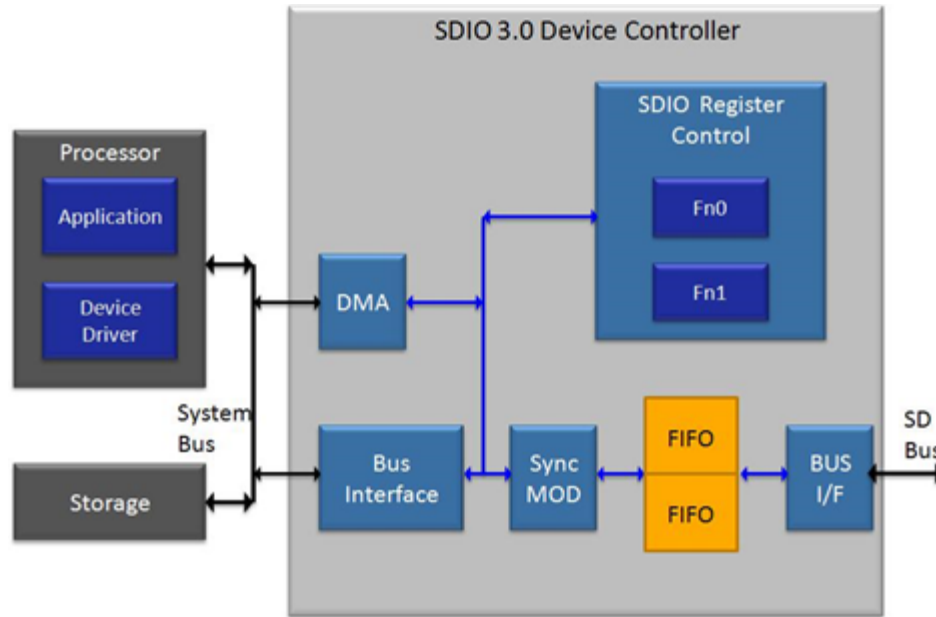


Figure 6: SDIO3.0 Device Controller Architecture

3.4.1 Functional Block Diagram Description

3.4.1.1 Bus Interface Unit (BIU)

The BIU communicates with the SD Host through the SD bus. SD1, SD4, SPI and 8-bit mode for embedded device are supported. The BIU houses the 16-bit CRC generator and checker for the data lines, 7-bit CRC generator and checker for the command and response lines, transmitter state machine, receiver state machine, interrupt state machine, BIU master state machine, command decoder and the response generator. The BIU bus capability is determined by bit values programmed in the R/W CCCR registers.

3.4.1.2 fn0_register

The fn0_register module contains the CIA (Common I/O Area). There are three distinct register structures supported within the CIA. They are Card Common Control Registers (CCCR), Function Basic Registers (FBR), Card Information Structure (CIS). The CIA is accessed by the Host via I/O reads and writes to function 0. The registers within the CIA also provided to enable/disable the operation of the I/O function(s), control the generation of interrupts. The registers in the CIA also provide information about the function(s) abilities and requirements.

3.4.1.3 fn_reg

The fn_reg module contains the function1 Host access registers. These registers are accessed by the Host through I/O reads and write to function 1. Whenever Interrupt asserted from the controller the Host should read these Registers to identify the source of interrupt. The Host can able to write these registers using cmd52 and cmd53. Host also uses these registers to send a message to ARM.

3.4.1.4 Synchronization Module (sync_mod)

The synchronization module has handshake logic to communicate with the BIU and on the other side communicates with the AHB Master and Slave Core.

3.4.1.5 FIFO Control

The FIFO control block contains one dual port FIFO for performing both read and write transactions. During write transaction (data transferred from Host to ARM), the data will be filled in to the first and second half of the FIFO alternatively. When data from first half of the FIFO is transferring to ARM, the second half of the FIFO will be filled and vice versa. The two FIFO's are alternatively used to store data which will give maximum throughput. During a read transaction (data transferred from ARM to Host) the data from ARM will be written in to the two half of the FIFO's alternatively. When data from one half of the FIFO is transferring to Host, the second half of the FIFO will be filled and vice-versa and thereby the throughput will be maximum.

3.4.1.6 Reg Control

The reg control block contains the function0, function1 ARM access registers. ARM Processor read/write these registers through AHB slave interface.

3.4.1.7 AHB Interface

The AHB Master is responsible for transferring data between the ARM Processor and the SDIO-AHB bridge for read and write operations using Scatter Gather DMA. AHB Slave Block houses the AHB slave interface signals. All the operational registers are in the reg control module. Reading and writing of these registers are handled by the SDIO-AHB bridge or ARM Processor. The ARM Processor should set the function ready bit in ESW Fun Ready register when it is ready to operate, indicating to the Host that all initialization has been done and Function is ready to operate.

3.4.1.8 ADMA Interface

The ADMA interface is responsible for controlling the ADMA transaction with system memory. It gives control signals to AHB Master Interface to initiate the AHB Bus transaction. This interface keeps performing ADMA fetch transaction with system memory whenever the described data length is over. It decodes the data address from the descriptor table and loads the same to AHB Master and also gets the data length from descriptor table and loads the number of bytes to AHB Master. If ADMA end bit is set in the descriptor table it will assert ADMA exhausted interrupt to ARM.

3.5 Signals

The Arasan SDIO Controller has four main interface groups.

1. SD Interface
2. AHB Slave Interface
3. AHB Master Interface
4. System Interface

Table 9: SD Interface Signals

Pin	Direction	Description
clk_sd	Input	SD Clock
clk_sd_inv	Input	Inverted SD Clock
SD_CMD	INOUT	SD4 bit mode: Command Line
		SD1 bit mode: Command Line
		SPI mode: Data Input
cmd_di	IN	Command Input
sel_sd_resp	OUT	Command Output
sel_resp_en	OUT	Command Output Enable
SD_DAT0	INOUT	SD4 bit mode: Data Line 0
		SD1 bit mode: Data Line
		SPI mode: Data Output
dat0_do	IN	Data0 Input
sel_tx_dat0	OUT	Data0 Output
sel_tx_dat0_en	OUT	Data0 Output Enable
SD_DAT1	INOUT	SD4 bit mode: Data Line1 or Interrupt (optional)
		SD1 bit mode: Interrupt
		SPI mode: Interrupt
dat1_irq	IN	Data1 Input
sel_tx_dat1	OUT	Data1 Output
sel_tx_dat1_en	OUT	Data1 Output Enable
SD_DAT2	INOUT	SD4 bit mode: Data Line2 or Read Wait (optional)
		SD1 bit mode: Read Wait (optional)
		SPI mode: Not Used
dat2_rw	IN	Data2 Input
sel_tx_dat2	OUT	Data2 Output
sel_tx_dat2_en	OUT	Data2 Output Enable
SD_DAT3	INOUT	SD4 bit mode: Data Line 3
		SD1 bit mode: Not Used
		SPI mode: Card Select
dat3_cs	IN	Data3 Input
sel_tx_dat3	OUT	Data3 Output
sel_tx_dat3_en	OUT	Data3 Output Enable

Table 10: AHB Slave Interface Signals

Signal	Direction	Description
clk_ahb	Input	AHB System Clock
ahb_sdio_haddr[16:0]	Input	Address Bus (Byte Addresses)
ahb_sdio_hwdata[31:0]	Input	Write Data Bus
ahb_sdio_hrdata[31:0]	Output	Read Data Bus
ahb_sdio_hwrite	Input	Write or Read Direction Indication
ahb_sdio_hsize[2:0]	Input	Size (Byte, Half Word or Word)
ahb_sdio_htrans[1:0]	Input	Transfer Type
ahb_sdio_hready_glb	Input	Global Ready
ahb_sdio_hready	Output	Slave Ready
ahb_sdio_hresp[1:0]	Output	Transfer Response
int_to_arm[30:0]	Output	Interrupt to Processor

Table 11: AHB Master Interface Signals

Signal	Direction	Description
ahb_sdio_dma_hbusreq	Output	AHB Bus request
ahb_sdio_dma_hgrant	Input	AHB Bus Grant
ahb_sdio_dma_haddr[31:0]	Output	AHB Slave DMA address (byte addresses)
ahb_sdio_dma_hwdata[31:0]	Output	AHB write data
ahb_sdio_dma_hrdata[31:0]	Input	AHB read data
ahb_sdio_dma_hwrite	Output	Write / Read Direction Indication
ahb_sdio_dma_hsize[2:0]	Output	Size (byte, half word or word)
ahb_sdio_dma_hburst[2:0]	Output	Burst Size
ahb_sdio_dma_hrdyglb	Input	Global ready signal
ahb_sdio_dma_htrans[1:0]	Output	Transfer type
ahb_sdio_dma_hresp[1:0]	Input	Transfer response

Table 12: System Interface Signals

Signal	Direction	Description
pwr_on_rst_n	Input	Active Low. Asynchronous Hardware reset from the External environment
ahb_clk_wkup	Output	Active high signal to wakeup AHB clock. At power-on-reset this signal will be asserted to 1'b1 as the chip is in manual mode when manual clk enable bit is set to 1'b1 in Clock wakeup register. When auto clk enable bit in Clock wakeup register is 1'b1, SDIO controller will enable the ahb_clk_wkup depending on the activity on the bus when required
pullup_en	Output	Pull-up Enable from the Arasan SDIO-AHB bridge for Card Detection
scan_mode	Input	Active Low. If set as 1'b1 - bypass all reset pulses generated internally 1'b0 - POR will be asserted asynchronously.

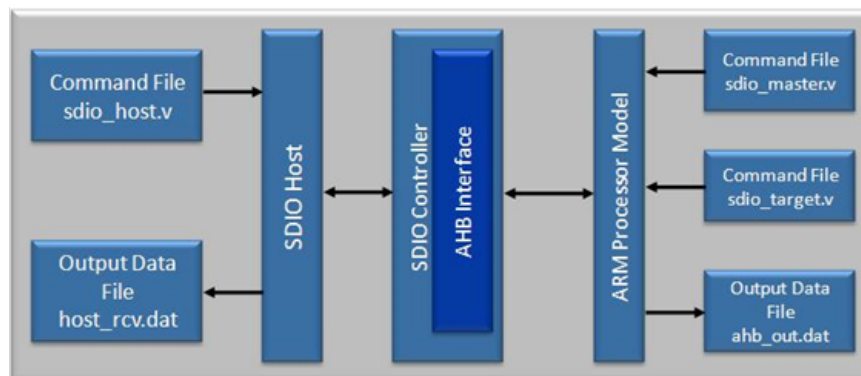
3.6 SoC Level Integration

3.6.1 IP Deliverables

- Verilog HDL of the IP core
- Test environment and test scripts
- Synthesis scripts
- User Guide

3.6.2 Verification Environment

The SDIO core has been verified in the simulation environment using the behavioral models of the surrounding environment (RTL verification). The RTL verification environment for SDIO core consists of the behavioral models to emulate the SD host and ARM processor interface.

**Figure 7: Verification Environment of SDIO Controller IP with AHB System Bus**

4 SD/SDIO 3 Device Controller IP

4.1 Overview

The Arasan SD-Combo-AHB Controller is a full / high speed card controller suitable for Memory cards and I/O Card applications like WLAN, Bluetooth with low power consumption for mobile operated devices. The controller supports SPI, 1-bit SD, 4-bit SD and 8-bit bus mode for embedded device. The Arasan SD-Combo-AHB Controller has an AHB interface, which allows the ARM Processor to configure the operational registers residing inside the AHB Slave core.

The CIS and CSA area is located inside the internal memory of ARM subsystem. The SDIO Registers (CCCR and FBR) and SD Registers (CSD, Password) are programmed by both the SD Host (through the SD Bus) and ARM Processor (through the AHB bus) via Operational registers.

When SD Card inserted into an SDIO aware Host, the detection of the card will be via the normal means described in Fig. 1. During the normal initialization and interrogation of the card by the Host, the card will identify itself as an SDIO device. The Host software will obtain the card information in a tuple (linked list) format and determine if that card's I/O function(s) are acceptable to activate. If the Card is acceptable, it will be allowed to power up fully and start the I/O function(s) built into it.

The SD memory card communication is based on an advanced 9-pin interface (Clock, Command, 4xData and 3 Power lines) designed to operate in different frequency depends on the corresponding mode of operation (SD3.0).

AHB is a new generation of AMBA bus, which is intended to address the requirements of high-performance synthesizable designs. AMBA AHB is a new level of bus which sits above the APB and implements the features required for high performance, high clock frequency systems including burst transfers, single clock edge operation etc. Before an AMBA AHB transfer can commence the bus master must be granted access to the bus. This process is started by the master asserting a request signal to the arbiter. Then the arbiter indicates when the master will be granted use of the bus. A granted bus master starts an AMBA AHB transfer by driving the address and control signals.

4.2 Features

- Meets SDIO card specification version 3.0.
- Meets SD Memory Card Physical Layer Specification version 3.0.
- Meets SD memory Card Security Specification Part 3.
- Meets AMBA 3.0 Specification
- Bus Master with Scatter Gather DMA
- Host clock rate varies between 0 and 208 MHz
- All SD bus modes supported including SPI, 1 and 4 bit SD and 8 bit for embedded device.
- Allows card to interrupt Host in SPI, 1 and 4 bit SD modes.
- Upto 382Mbits per second read and write rates using 4 parallel data lines

- Cyclic Redundancy Check CRC7 for command and CRC16 for data integrity-CRC checking optional in
- SPI mode
- Programmable through a standard AHB Slave interface
- Application specific Commands
- Comfortable Erase Mechanism
- Card removal during read operation will never harm the content.
- Password protection of Cards
- Write Protect feature using mechanical switch
- Built-in write protection features (permanent and temporary)
- Card Detection (Insertion / Removal)
- Switch function command supports Bus Speed Mode, Command System, Drive Strength, and future functions.
- Card responds to Direct read/write (IO52) and Extended read/write (IO53) transactions.
- Supports Read wait Control operation.
- Supports Suspend/Resume operation.
- Capacity of Memory
 - Standard Capacity SD Memory Card (SDSC) : Up to and including 2GB.
 - High Capacity SD Memory Card (SDHC) : More than 2GB and up to and including 32GB.
 - Extended Capacity SD Memory Card (SDXC) : More than 32GB and up to and including 2TB.
- Bus Speed Mode (using 4 parallel data lines)
 - Default Speed mode : 3.3V signaling, Frequency up to 25 MHz, up to 12.5 MB/sec
 - High Speed mode : 3.3V signaling, Frequency up to 50 MHz, up to 25 MB/sec
 - SDR12: 1.8V signaling, Frequency up to 25 MHz, up to 12.5 MB/sec
 - SDR25: 1.8V signaling, Frequency up to 50 MHz, up to 25 MB/sec
 - SDR50: 1.8V signaling, Frequency up to 100 MHz, up to 50 MB/sec
 - SDR104: 1.8V signaling, Frequency up to 208 MHz, up to 104 MB/sec
 - DDR50: 1.8V signaling, Frequency up to 50 MHz, sampled on both clock edges, up to 50MB/sec

4.3 Bus Topology

The SD Card defines two alternative communication protocols SD and SPI. Applications can choose either one of modes. UHS-I cards does not support SPI mode and SD 1 bit mode. Mode selection is transparent to the Host. After reset command the Card automatically moves to default mode and will expect all further communication to be in the same mode. Therefore, applications which uses only one communication mode do not have to be aware of the other.

SD bus allows dynamic configuration of the number of data lines. After power up by default, the SD card will use one DAT0 for data transfer. After initialization the Host can change the bus width

(number of active data lines). This feature allows easy trade off between HW cost and system performance.

8 bit bus mode can be supported only for and Embedded device. Achieving higher performance in the lower frequency is the objective of this mode.

4.3.1 SD-Combo Card Initialization Sequence

The SD-Combo card initialization sequence is described below for SD mode and SPI mode.

Note: To access the card being already in inactive state, a hard reset must be done by switching the power supply off and on.

IORDY time-out: Timeout for Initialization process is 1 sec.

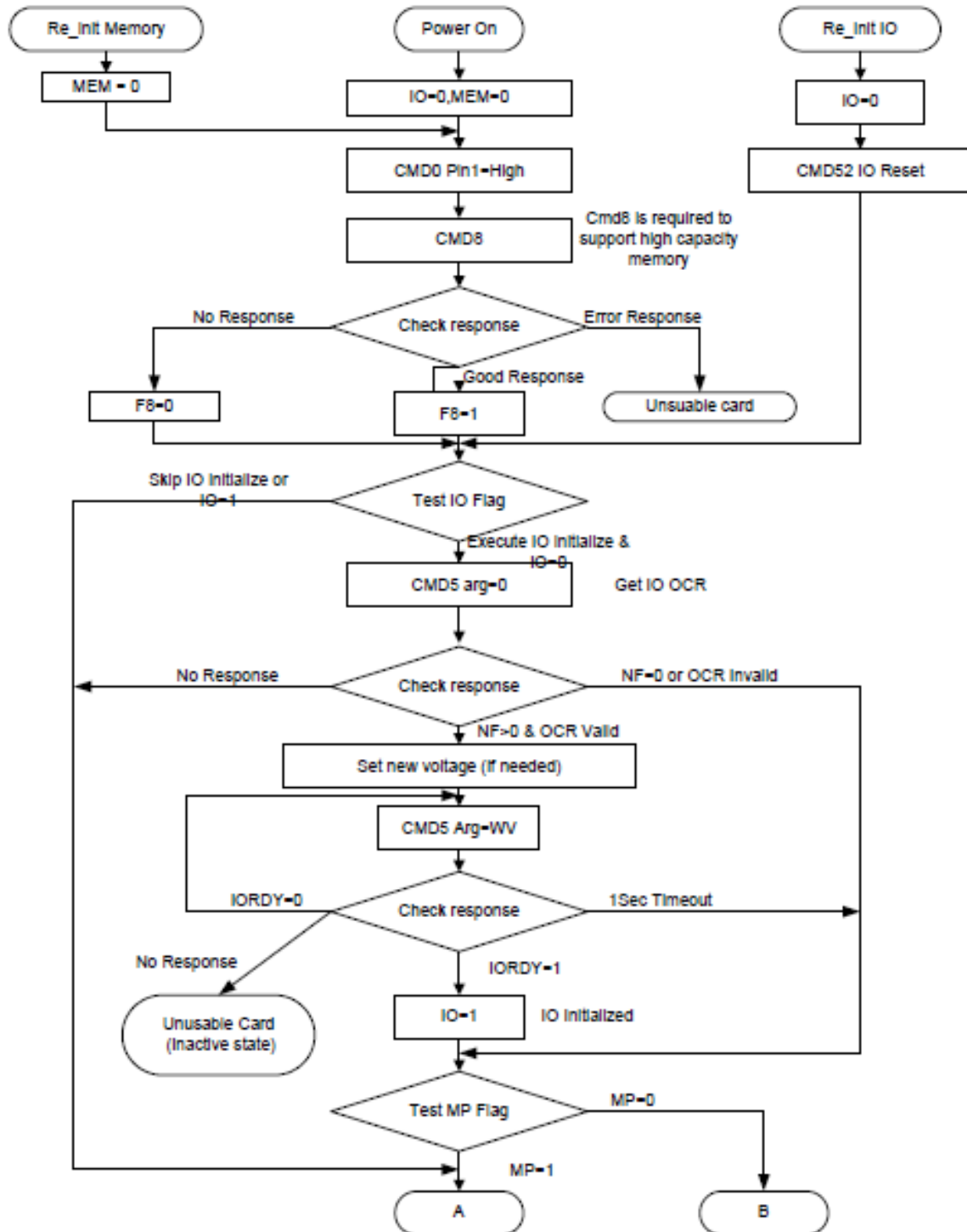


Figure 8: SD Combo card initialization flow in SD mode

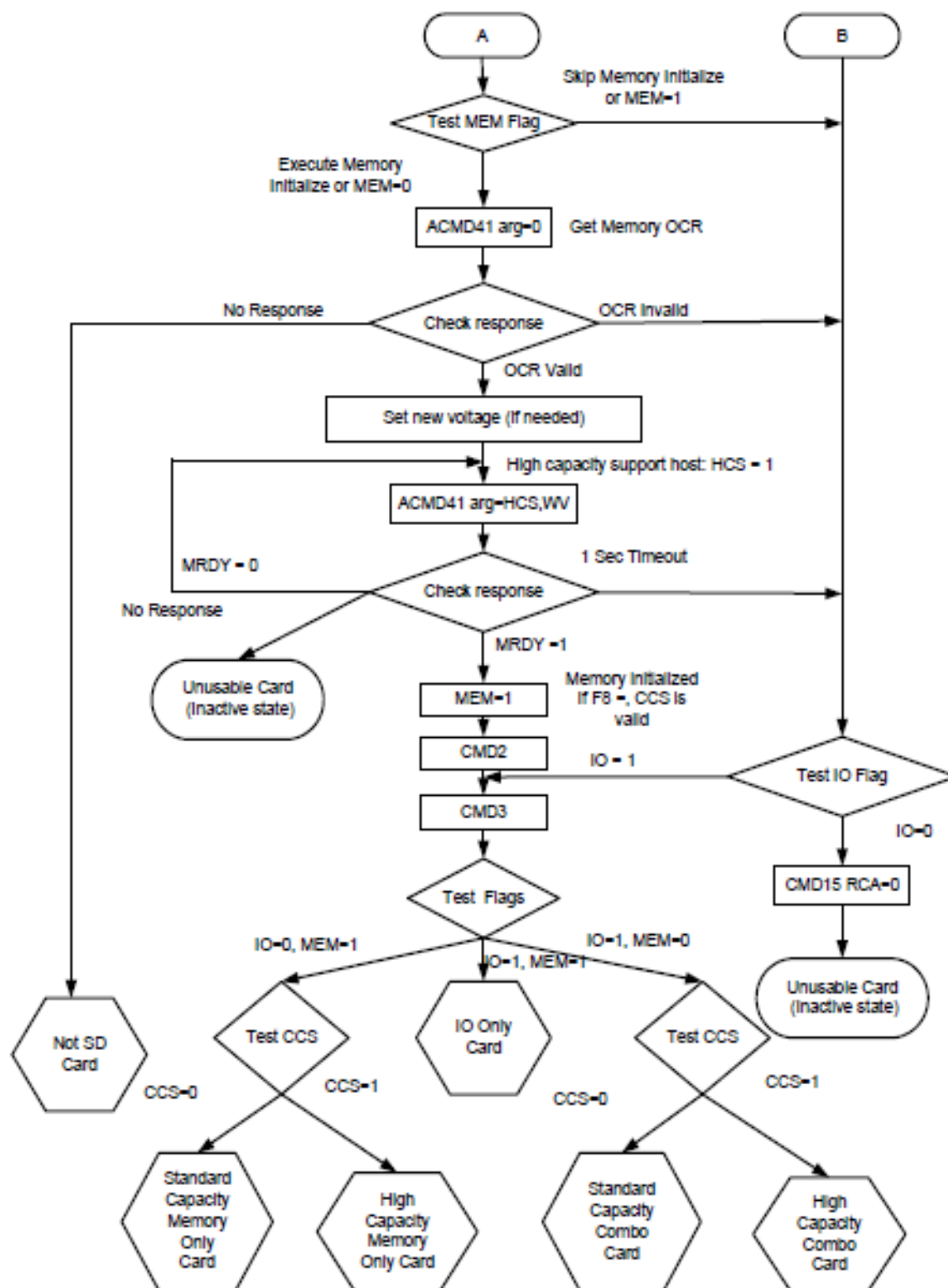


Figure 9: SD Combo card initialization flow in SD mode Continued

- Variables:
 - NF: Number of I/O Functions (CMD5 Response)
 - MP: Memory present Flag (CMD5 Response)
 - IORDY: I/O Power-up Status (C bit in the CMD5 response)
 - MRDY: Memory Power-up Status (OCR Bit31)
 - HCS: Host Capacity Support (ACMD41 Argument)
 - CCS: Card Capacity Status (ACMD41 Response)
- Flags:
 - IO: I/O Functions Initialized Flag
 - MEM: Memory Initialized Flag
 - F8: CMD8 Flag

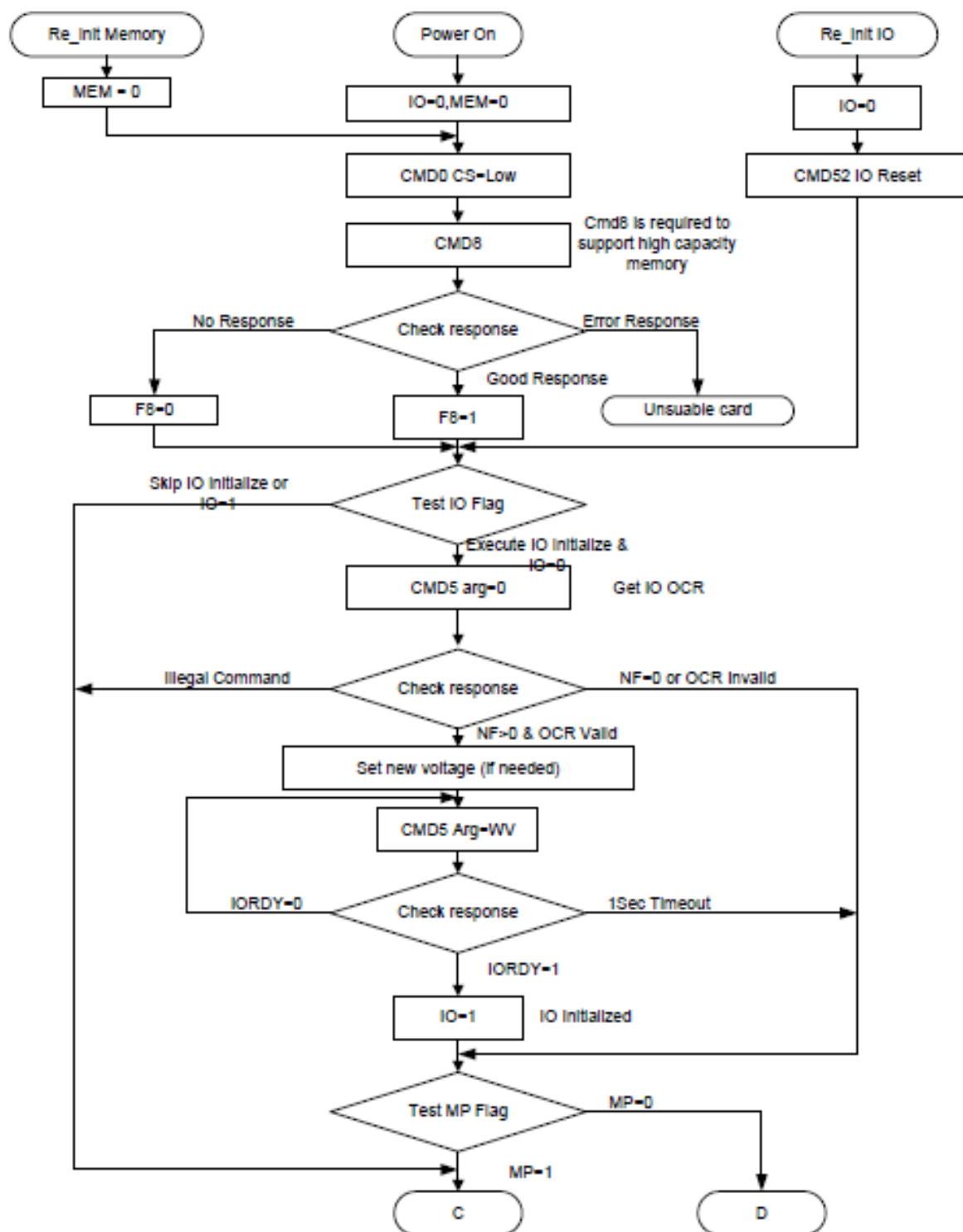


Figure 10: SD Combo card Initialization flow in SPI mode

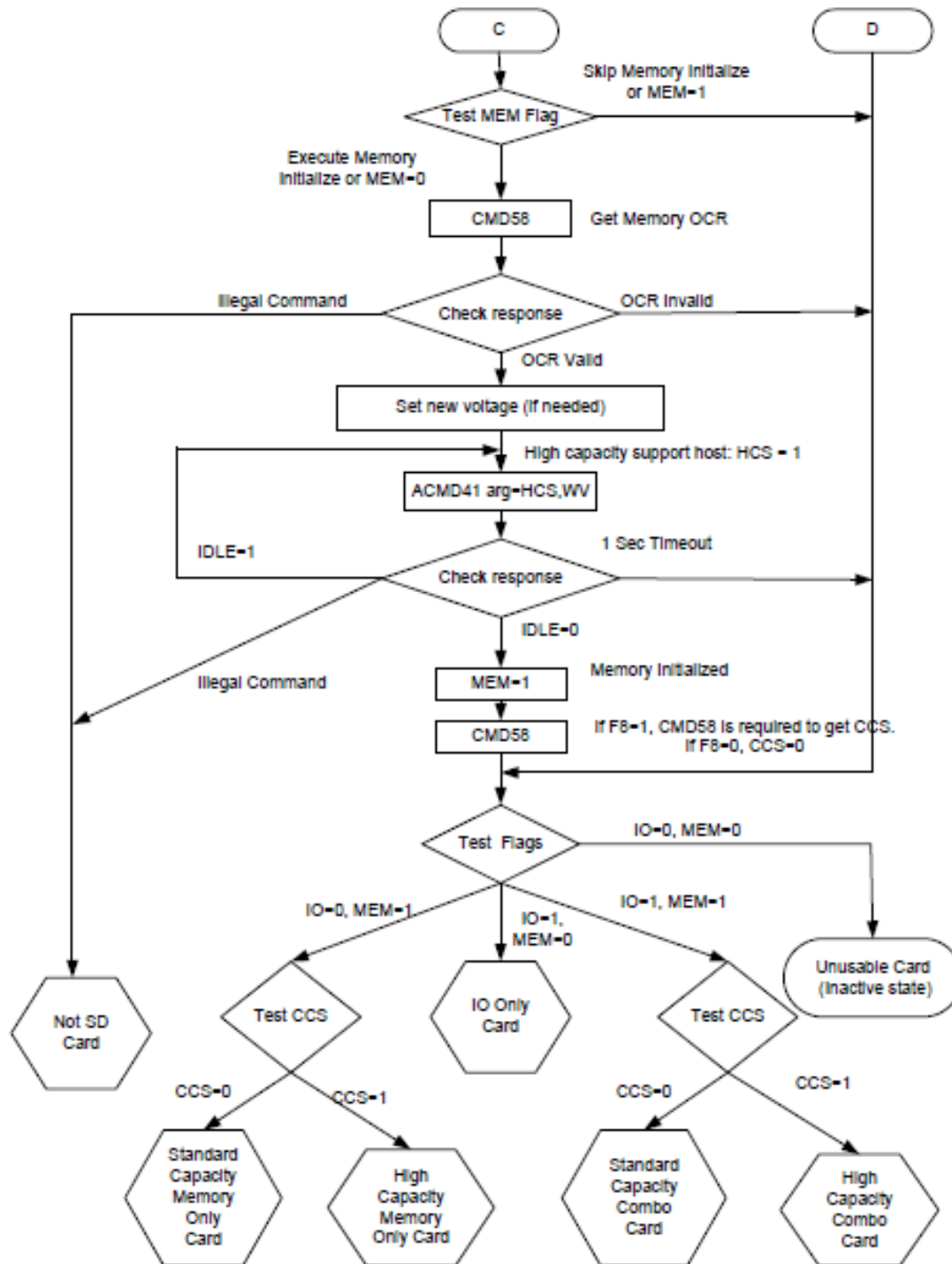


Figure 11: SD Combo card Initialization flow in SPI mode Continued

- Variables:
 - NF: Number of I/O Functions (CMD5 Response)
 - MP: Memory present Flag (CMD5 Response)
 - IORDY: I/O Power-up Status (C bit in the CMD5 response)

- MRDY: Memory Power-up Status (OCR Bit31)
- HCS: Host Capacity Support (ACMD41 Argument)
- CCS: Card Capacity Status (ACMD41 Response)
- Flags:
 - IO: I/O Functions Initialized Flag
 - MEM: Memory Initialized Flag
 - F8: CMD8 Flag

4.4 Architecture

4.4.1 Functional Block Diagram

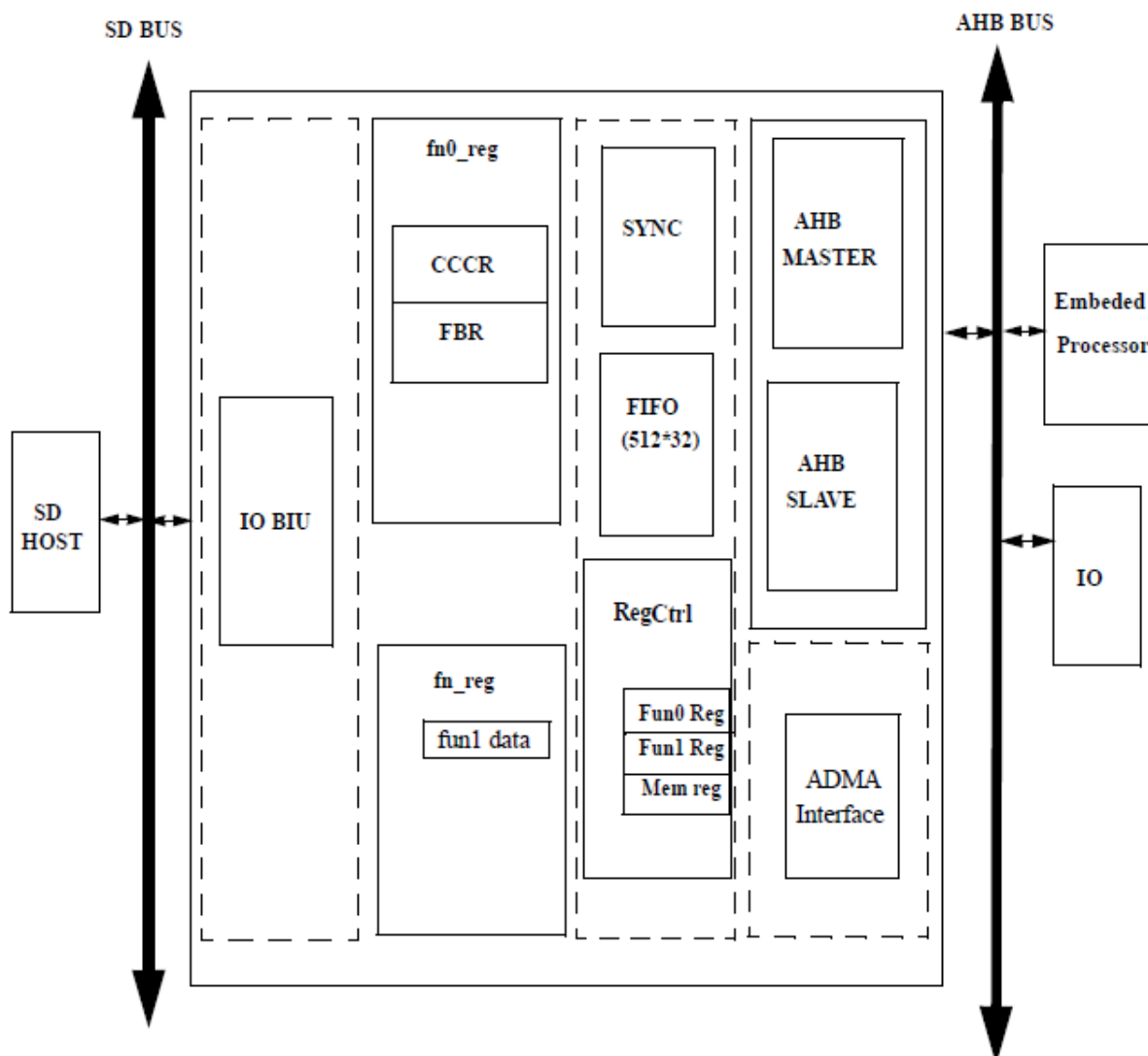


Figure 12: SD_Combo_AHB Architecture

4.4.2 Functional Block Diagram Description

4.4.2.1 Bus Interface Unit (BIU)

The BIU communicates with the SD Host through the SD bus. SD1, SD4, SPI and 8 bit mode for Embed-ded device are supported. The BIU houses the 16 bit CRC generator and checker for the data lines, 7 bit CRC generator and checker for the command and response lines, transmitter state machine, receiver state machine, interrupt state machine, BIU master state machine, command decoder, and the response genera-tor. The BIU bus capability is determined by bit values programmed in the R/W CCCR registers. The Mem BIU bus capability is determined when the Mem Card registers are programmed with appropriate bit values.

4.4.2.2 fn0_register

The fn0_register module contains the CIA (Common I/O Area). There are three distinct reigister structures supported within the CIA. They are Card Common Control Registers (CCCR), Function Basic Registers (FBR), Card Information Structure (CIS). The CIA is accessed by the Host via I/O reads and writes to function 0. The registers within the CIA also provided to enable/disable the operation of the I/O function(s), control the generation of interrupts. The registers in the CIA also provide information about the function(s) abilities and requirements.

4.4.2.3 fn_register

The fn_reg module contains the funcion1 Host access registers. These registers are accessed by the Host via I/O reads and writes to function 1. Whenever Interrupt asserted from the controller the Host should read these Registers to identify the source of interrupt. The Host can able to write these registers using cmd52 and cmd53. Host can also use these registers to send a message to ARM.

4.4.2.4 Synchronization module (sync_mod)

This block has handshake logic to communicate with the BIU and on the other side communicates with the AHB Master and Slave Core.

4.4.2.5 Fifo Control

This block contains one dual port fifo for performing both read and write transactions. During write trans-action (data transferred from Host to ARM), the data will be filled in to the first and second half of the fifo alternatively. When data from first half of the fifo is transferring to ARM, the second half of the fifo will be filled and vice versa. The two fifo's are alternatively used to store data which will give maximum through-put. During a read transaction (data transferred to ARM to Host) the data from ARM will be written in to the two half of the fifo's alternatively. When data from one half of the fifo is transferring to Host, the second half of the fifo will be filled and vice-versa and thereby the throughput will be maximum.

Note: FIFO depth can be varied using parameter defined in ram_params.v based on that maximum block size supported will change. If the maximum block size supported is 1K then the fifo size should be 2K. Two 1k fifo's are used to support ping pong mechanism.

4.4.2.6 Reg Control

The Reg control block contains the function0, function1, memory ARM access registers. ARM Processor read/write these registers through AHB slave interface.

4.4.2.7 AHB interface

The AHB Master is responsible for transferring data between the ARM Processor and Arasan SD-Combo-AHB Bridge for read and write operations using Scatter Gather DMA. AHB Slave Block houses the AHB slave interface signals. All the operational registers are in the reg control module. Reading and writing of these registers are handled by SD-Combo-AHB Bridge or the ARM Processor. The ARM Processor should set the function ready bit in ESW Fun Ready register when it is ready to operate, indicating to the SD Host that all initialization has been done and Function is ready to operate.

4.4.2.8 ADMA interface

This ADMA interface is responsible for controlling the ADMA transaction with system memory. It gives control signals to AHB Master Interface to initiate the AHB Bus transaction. This interface is keep on per-forming adma fetch transaction with system memory whenever the described data length is over. It decodes the data address from the descriptor table and load the same to AHB Master and also gets the data length from descriptor table and load the no of bytes to AHB Master. If adma end bit is set in the descriptor table it will assert adma exhausted interrupt to ARM.

4.5 Signals

4.5.1 Signal Interfaces

The Arasan SDIO Combo Controller has four main interface groups,

1. SD Interface
2. AHB Slave Interface
3. AHB Master Interface
4. System interface

Table 13: SD Interface Signals

Pin	Direction	Description
clk_sd	Input	SD Clock
clk_sd_inv	Input	Inverted SD Clock
SD_CMD	INOUT	SD4 bit mode: Command Line
		SD1 bit mode: Command Line
		SPI mode: Data Input
cmd_di	IN	Command Input
sel_sd_resp	OUT	Command Output
sel_resp_en	OUT	Command Output Enable
SD_DAT0	INOUT	SD4 bit mode: Data Line 0
		SD1 bit mode: Data Line
		SPI mode: Data Output
dat0_do	IN	Data0 Input
sel_tx_dat0	OUT	Data0 Output
sel_tx_dat0_en	OUT	Data0 Output Enable
SD_DAT1	INOUT	SD4 bit mode: Data Line1 or Interrupt (optional)
		SD1 bit mode: Interrupt
		SPI mode: Interrupt
dat1_irq	IN	Data1 Input
sel_tx_dat1	OUT	Data1Output
sel_tx_dat1_en	OUT	Data1 Output Enable
SD_DAT2	INOUT	SD4 bit mode: Data Line2 or Read Wait (optional)
		SD1 bit mode: Read Wait (optional)
		SPI mode: Not Used
dat2_rw	IN	Data2 Input
sel_tx_dat2	OUT	Data2 Output
sel_tx_dat2_en	OUT	Data2 Output Enable
SD_DAT3	INOUT	SD4 bit mode: Data Line 3
		SD1 bit mode: Not Used
		SPI mode: Card Select
dat3_cs	IN	Data3 Input
sel_tx_dat3	OUT	Data3 Output
sel_tx_dat3_en	OUT	Data3 Output Enable

Table 14: AHB Slave Interface Signals

Signal	Direction	Description
clk_ahb	Input	AHB System Clock
ahb_sdio_haddr[16:0]	Input	Address Bus (Byte Addresses)
ahb_sdio_hwdata[31:0]	Input	Write Data Bus
ahb_sdio_hrdata[31:0]	Output	Read Data Bus
ahb_sdio_hwrite	Input	Write or Read Direction Indication

ahb_sdio_hsize[2:0]	Input	Size (Byte, Half Word or Word)
ahb_sdio_htrans[1:0]	Input	Transfer Type
ahb_sdio_hready_glb	Input	Global Ready
ahb_sdio_hready	Output	Slave Ready
ahb_sdio_hresp[1:0]	Output	Transfer Response
int_to_arm[30:0]	Output	Interrupt to Processor

Table 15: AHB Master Interface Signals

Signal	Direction	Description
ahb_sdio_dma_hbusreq	Output	AHB Bus request
ahb_sdio_dma_hgrant	Input	AHB Bus Grant
ahb_sdio_dma_haddr[31:0]	Output	AHB Slave DMA address (byte addresses)
ahb_sdio_dma_hwdata[31:0]	Output	AHB write data
ahb_sdio_dma_hrdata[31:0]	Input	AHB read data
ahb_sdio_dma_hwrite	Output	Write / Read Direction Indication
ahb_sdio_dma_hsize[2:0]	Output	Size (byte, half word or word)
ahb_sdio_dma_hburst[2:0]	Output	Burst Size
ahb_sdio_dma_hrdyglb	Input	Global ready signal
ahb_sdio_dma_htrans[1:0]	Output	Transfer type
ahb_sdio_dma_hresp[1:0]	Input	Transfer response

Table 16: System Interface Signals

Signal	Direction	Description
pwr_on_rst_n	Input	Active Low. Asynchronous Hardware reset from the External environment
ahb_clk_wkup	Output	Active high signal to wakeup AHB clock. At power-on-reset this signal will be asserted to 1'b1 as the chip is in manual mode when manual clk enable bit is set to 1'b1 in Clock wakeup register. When auto clk enable bit in Clock wakeup register is 1'b1, SDIO controller will enable the ahb_clk_wkup depending on the activity on the bus when required
pullup_en	Output	Pull-up Enable from the Arasan SDIO-AHB bridge for Card Detection
scan_mode	Input	Active Low. If set as 1'b1 - bypass all reset pulses generated internally 1'b0 - POR will be asserted asynchronously.

4.6 SoC Level Integration

4.6.1 IP Deliverables

- Verilog HDL of the IP core
- Test environment and test scripts
- Synthesis scripts
- Sample ARM firmware and software drivers
- User Guide

5 eMMC 4.51 Device Controller IP

5.1 Overview

Arasan's eMMC 4.51 Device Controller IP is compliant with JEDEC JESD84-B451 standard. The controller provides a bandwidth of up to 1.6 Gbps in 200 MHz mode. A NAND Flash memory device can be connected to the eMMC Device controller. In such an implementation, the controller's AHB interface provides a channel for data transfers between the Device Controller and a NAND flash controller (also available from Arasan). The Device Controller supports the newer eMMC functions such as E2MMC devices, extended partitioning, command packing, context IDs, data tags and dynamic device capacity. The memory controller operates at a maximum frequency of 200 MHz. The interface supports MMC 1-bit, 4-bit and 8-bit modes.

eMMC supports power-on booting without the upper level of software driver which simplifies system design. The controller shields the host system from the functional differences amongst various NAND flash architectures (such as MLC). The explicit sleep mode allows the host to instruct the controller to directly enter a low power sleep mode. The controller supports block lengths or sector sizes of 512, 1024, 2048 and 4096 bytes. The eMMC 4.51 added Extended Security Commands and re-introduced Secure Erase and Secure Trim.

5.2 Features

- Complies with JESD84-B451 Embedded Multimedia Card, Electrical Standard (4.51 Device)
- Supports high speed interface timing mode of up to 200 MB/s single data rate bus (HS200)
- Supports tuning concept for read operation
- Supports DDR interface with data transfer rate of 104 MB/s
- Supports new H/W Reset pin
- Supports Power-On Write Protection groups, Temporary and Permanent Write Protection features
- Supports Alternative Boot operation mode
- Supports Write Protection features on H/W Reset
- Supports Replay Protected Memory Block (RPMB) and access control
- Supports partition management by defining general purpose partition with enhancement attribute features
- Supports Write Protection on Boot area
- Supports Boot operation in High Speed and DDR mode
- Cyclic Redundancy Check CRC7 for command and CRC16 for data integrity
- Full backward compatibility with previous multimedia card systems (1-bit data bus, multi card systems)
- Supports general purpose R/W command
- Size of eMMC device capacity can be more than 2GB
- Supports block lengths of 512 bytes and sector size of 4K bytes for high capacity

- Error injection capabilities
 - Supports Boot operation mode in simple boot sequence method
 - Supports Sleep mode for power saving
 - Supports Reliable Write operation that performs safe-updates on sudden power failure
 - Supports Packed command flow sequence
 - Supports Power-Off notification flow
 - Supports Discard and Sanitize command
 - Supports context IDs and data tag mechanism
 - Supports security protocol write and read command
 - Supports secure erase and secure trim
- AMBA Compliance:
 - Complies with AMBA specification version 2.0
 - Supports Incremental Burst Transfers on DMA mode
 - Supports Register Transfer on non DMA mode
 - Supports Retry and Split
 - Speed Classes Supported:
 - Arasan's IP supports MMC speed classes: 0, 2, 4, 5, 6, 7, 8, 9, 10.
 - Card Enumeration:
 - The Arasan eMMC Device IP is enumerated by a host. Card insertion or removal in the slot is signaled to the host. The type of the card and supported memory size are informed to the host controller via CID, CSD, Ext_CSD registers housed inside the Arasan eMMC Device IP.

5.3 Architecture

5.3.1 Functional Description

The Arasan eMMC 4.51 device IP provides a solution to integrate the controller with memory. This IP core is fully tested to meet the requirements of the JESD84-B451 Multi-Media Card Electrical Standard.

The controller's interfacing pins support MMC 1-bit, 4-bit and 8-bit modes. The controller portion of the core operates at eMMC clock provided by an eMMC Host that can vary from 0 to 200 MHz.

The eMMC device is accessed using a 9-pin or 13-pin interface (clock, command, 4 or 8 data pins, data strobe, 3 power pins) with an option for an additional Reset pin. It is designed to operate at a maximum operating frequency of 200 MHz and supports high speed interface timing mode of up to 400 MB/s. eMMC supports power-on booting without the upper level of software driver. The explicit sleep mode allows the host to instruct the controller to directly enter the sleep mode. The functional block diagram of Arasan eMMC 4.51 device IP is shown in the following figure.

5.3.2 Functional Block Diagram

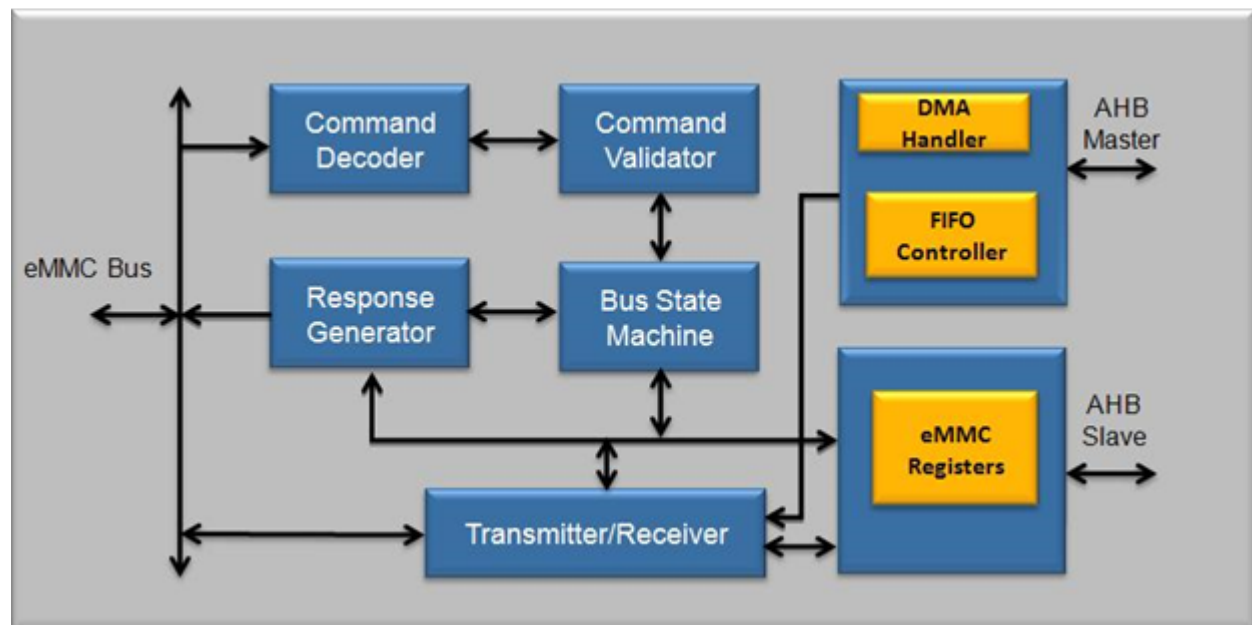


Figure 13: eMMC 4.51 Device Controller Functional Block Diagram

5.3.3 Functional Block Diagram Description

5.3.3.1 Clocks

The IP operates on MMC_clk provided by the MMC host that ranges from 0-200 MHz. AHB clock is used by the IP for data transfer and programming operations.

5.3.3.2 Reset

Power-on reset is generated externally upon sensing the VDD signals. This Power-on reset is ORed with the Soft_Reset and results in the Reset signal to all the blocks inside the controller. The Soft Reset is generated whenever a GO_IDLE_STATE [CMD0] without argument is sent by the host.

5.3.3.3 Protocol Layer

This layer comprises of command decoder and validator, response generator, receiver, and transmitter sections.

5.3.3.4 Command Decoder [cmd_dec]

The Command Decoder registers the 48-bit command for all modes (MMC1_bit, MMC4_bit and MMC8_bit). This block performs cyclic redundancy checks (CRC) on the received commands.

5.3.3.5 Response Generator [resp_gen]

The Response Generator sends appropriate responses for the commands for all SD/ MMC modes.

5.3.3.6 Transmitter/ Receiver [tx_rx]

The Transmitter/Receiver handles the data transactions for all SD/MMC modes. This block controls the data lines of the MMC/SD bus.

5.3.3.7 Command Validator [cmd_valdr]

All commands received by eMMC devices are validated based on the eMMC's state. This Command Validator checks for parameter errors, address errors, and all the errors in the argument field of the command. This block generates the control for the resp_gen block.

5.3.3.8 Bus State Machine

This block handles SD and eMMC bus states as mentioned in the eMMC specification and SD physical specification.

5.3.3.9 eMMC 4.51 Register Sets/ Operational Registers

This block holds the OCR, CID, CSD, SCR, EXT_CSD and RCA for the SD or eMMC devices. These register contents are written using AHB Slave Writes.

5.3.3.10 RAM

Two 128 x32-bit Dual-port RAMs (DPRAM) are used as sector buffers to support 512-byte block size. A 64 x 32-bits Dual-Port RAM used for storing some of the contents of the Operational Registers.

5.3.3.11 DMA Handler

The DMA Handler contains a FIFO and initiates DMA Read or Write operations based on the interrupt serviced by the ARM processor.

5.4 Signal Interfaces

The Arasan eMMC 5.0 Device IP has five main interface groups:

- eMMC Bus Interface Pins
- AHB Target Interface Signals
- AHB Master Interface Signals
- Two 128x32 Dual-Port RAM Interface Signals
- 64x32 RAM Interface Signals

Note: Active low signals have a suffix of '_n'

Table 17: eMMC Bus Interface Signals

Pin	Direction	Description
mmc_clk	Input	mmc clock (0-52 MHz)
pow_on_rst	Input	VDD line
cmd_in	Input	Command line input
data_in[7:0]	Input	Data bus input
rst_n	Input	Hardware Reset pin
cmd_en	Output	Command line enable
data_out[7:0]	Output	Data bus output
cmd_out	Output	Command line output
data_out_en[7:0]	Output	Data bus enable
pull_up_en_dat[7:0]	Output	This output signal is used to connect or disconnect the pull-up for the data lines
exit_low_power	Output	Trigger signal to exit low power mode

Table 18: AHB Target Interface Signals:

Pin	Direction	Description
hclk	Input	This clock times all bus transfers. All signal timings are related to the rising edge of hclk
hrst_n	Input	The bus reset signal is active low and is used to reset the system and the bus. This reset signal is asserted asynchronously and de-asserted synchronously after the rising edge of hclk
hrdy_in	Input	External Master ready. This signal is routed to the AHB target core
tar_haddr[31:0]	Input	The 32-bit system addresses bus. This signal is routed to the AHB target core.
tar_htrans[1:0]	Input	Indicates the type of the current transfer which can be non-sequential, sequential, idle or busy
tar_hsize[2:0]	Input	Indicates the size of the transfer which is typically byte, half word or word
tar_hburst[2:0]	Input	Indicates if the transfer forms part of a burst
tar_hwdata[31:0]	Input	The write data bus is used to receive data from the master during write operations
tar_hwrite	Input	When high, this signal indicates a write transfer and when low a read transfer
tar_hsel	Input	When high, this signal selects the target and when low de-selects target
tar_hr_data[31:0]	Output	The read data bus is used to transfer data to the bus master during read operations
tar_hrdy	Output	When high, this signal indicates that a transfer has finished on the bus. This signal is driven low to extend a transfer
tar_hrsp[1:0]	Output	The transfer response provides additional information on the status of a transfer whether it is okay, error, retry or split
tar_intr	Output	Processor interrupt

Table 19: AHB Master Interface Signals

Pin	Direction	Description
mas_hb_req	Output	AHB bus request
mas_hgrant	Input	AHB bus grant
mas_haddr [31:0]	Output	Bus address (byte address)
mas_hwdata[31:0]	Output	The write data bus is used to transfer data to the bus slaves during write operations
mas_hrdata [31:0]	Input	The read data bus is used to receive data from bus slaves during read operations
mas_hwrite	Output	When high, this signal indicates a write transfer and

Pin	Direction	Description
		when low a read transfer
mas_hsize [2:0]	Output	Indicates the size of the transfer which is typically byte, half word or word
mas_htrans [1:0]	Output	Indicates the type of the current transfer which can be non-sequential, sequential, idle or busy
mas_hburst [2:0]	Output	Indicates if the transfer forms part of a burst. Supports four beat incremental burst.
mas_hrdy	Input	When high, this signal indicates that a transfer has finished on the bus. This signal is used to extend a transfer when it is low.
mas_hrsp [1:0]	Input	The transfer response indicates additional information on the status of a transfer whether it is okay, error, retry or split

Notes:

- Arasan SD/eMMC device IP supports two types of Data Packet format.
 - Usual data (8-bit width): The usual data (8-bit width) are sent in LSB (Least Significant Byte) first, MSB (Most Significant Byte) last sequence. But in the individual byte, it is MSB (Most Significant Bit) first, LSB (Least Significant Bit) last.
 - Wide width data (SD Memory Register): The wide width data is shifted from the MSB bit
- HPROT signal is not used.

Table 20: 128x32 Dual-Port RAM1 Interface Signals

Pin	Direction	Description
sd_dat_out1[31:0]	Input	SD data output from the RAM during SD read
ahb_dat_out1[31:0]	Input	AHB data output from the RAM during AHB read
sd_cs1	Output	Active low chip select for SD
sd_wren1	Output	Active low write enable signal for SD write
sd_addr1 [6:0]	Output	SD read/write address
sd_dat_in1 [31:0]	Output	SD data input to the RAM during SD write
sd_out_en1	Output	Active low output enable signal for SD read
ahb_cs1	Output	Active low chip select for AHB
ahb_wren1	Output	Active low write enable signal for AHB write
ahb_addr1[6:0]	Output	AHB read/write address
ahb_dat_in1 [31:0]	Output	AHB data input to the RAM during AHB write
ahb_out_en1	Output	Active low output enable signal for AHB read

Table 21: 128x32 Dual-Port RAM2 Interface Signals

Pin	Direction	Description
sd_dat_out2[31:0]	Input	SD data output from the RAM during SD read
ahb_dat_out2[31:0]	Input	AHB data output from the RAM during AHB read
sd_cs2	Output	Active low chip select for SD
sd_wren2	Output	Active low write enable signal for SD write
sd_addr2 [6:0]	Output	SD read/write address
sd_dat_in2 [31:0]	Output	SD data input to the RAM during SD write
sd_out_en2	Output	Active low output enable signal for SD read
ahb_cs2	Output	Active low chip select for AHB
ahb_wren2	Output	Active low write enable signal for AHB write
ahb_addr2[6:0]	Output	AHB read/write address
ahb_dat_in2 [31:0]	Output	AHB data input to the RAM during AHB write
ahb_out_en2	Output	Active low output enable signal for AHB read

Note: This RAM module is used for the data transfer from and to the AHB during the DMA Read and Write operations.

Table 22: 128x32 RAM Interface Signals

Pin	Direction	Description
sd_ram_out[31:0]	Input	SD data output from the RAM during SD read
ahb_ram_op[31:0]	Input	AHB data output from the RAM during AHB read
ram_cs_en	Output	Active low chip select for SD
sd_ram_wr	Output	Active low write enable signal for SD write
sd_ram_addr [6:0]	Output	SD read/write address
sd_ram_in [31:0]	Output	SD data input to the RAM during SD write
ram_wr_data [31:0]	Output	AHB data input to the RAM during AHB write
ram_cs	Output	Active low chip select for AHB
ahb_ram_we	Output	Active low write enable signal for AHB write
ahb_ram_addr[6:0]	Output	AHB read/write address
out_en_ram_256	Output	Active low output enable signal for SD read
int_ram_out_en	Output	Active low output enable signal for AHB read

Note: This RAM is used to store the values of the SD/MMC card registers.

5.5 SOC Level Integration

5.5.1 Verification Environment

The eMMC 4.51 Device IP core has been verified in the simulation environment using the behavioral models of the surrounding environment (RTL verification).

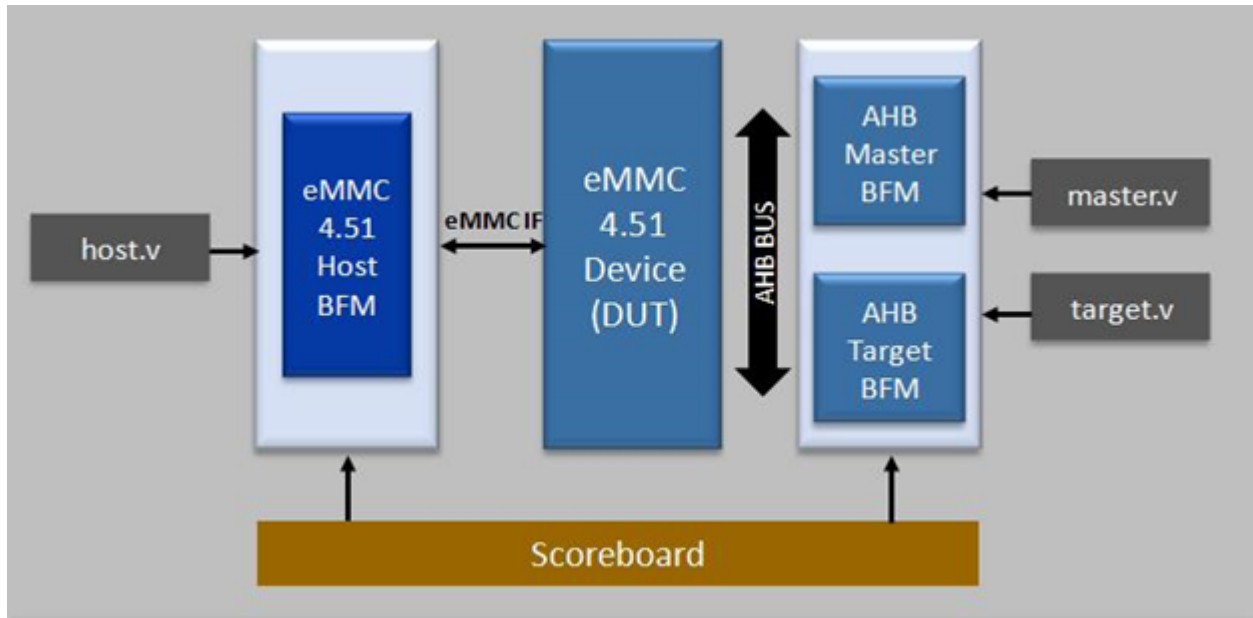


Figure 14: Verification Environment of eMMC 4.51 Device

5.5.2 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

5.5.3 IP Deliverables

- Verilog HDL
- Synthesis scripts
- Test Environment and test scripts
- User guide

6 Hardware Validation Platform

6.1 Overview

Designed to be cost effective and Linux based, the SD 3.0/eMMC 4.51 hardware validation platform (HVP) consists of Arasan's SD 3.0/eMMC 4.51 IP mapped into an FPGA, offering full speed physical connectivity to a complementary SoC host or memory card device. Arasan's active participation in the JEDEC/eMMC standards bodies and early customer engagement has made Arasan's SD/eMMC controller IP a "gold master" reference in the industry.

An equally important part of the SD 3.0/eMMC 4.51 HVP is the software stack, which abstracts all the low level software drivers and hardware to a loadable shared object with easy to use API's. Hence, an applications software, validation or systems engineer does not need to delve into the protocol and signaling details, and considers the Arasan platform as a programmable complementary device. The SD 3.0/ eMMC 4.51 HVP can be used by system developers, system integrator, software developers and system quality analysts to debug as well as validate their products during the product life-cycle.

6.2 Features

- Specification Compliance:
 - Supports SD host controller standard specification Version 3.0
 - Supports SD Physical Specification Version 3.0
 - Supports SDIO Card Specification Version 3.0
 - Supports Embedded SD Physical Specification Version 2.1
 - Meets eMMC specification Version 4.2/4.3/4.4/4.15/5.0
- SDIO/SD/SDHC/SDXC/eSD Device Validation:
 - Card detection (Insertion / Removal)
 - Host clock rate variable between 0 and 208 MHz
 - Supports SD 1/4 bit SDIO modes
 - Supports both 3.3V and 1.8V SD interface signals
 - CRC7 and CRC16 Cyclic Redundancy Checks
 - Direct and extended read/write (IO52, IO53) transactions
 - Supports read wait control operation
 - Suspend/resume operation
 - Multi block read and write and PCIe interrupt
- eMMC Device Validation
 - HS400 mode with 200 MHz DDR operation
 - Field firmware update, eMMC device health report, eMMC production state awareness, Secure removal types

- CRC7 and CRC16 Cyclic Redundancy Checks
- eBoot read on power-on
- eBoot read/write in boot partitions after card initialization
- eBoot & Normal Read/Write access in 1/4/8 bit wide modes
- Alternate Boot Operation Mode

6.3 HVP Architecture

The SD/SDIO/eMMC 4.51 Hardware Validation Platform (HVP) is a complete Linux-based system for validation of SD, SDIO and eMMC compliant devices. It may be used for SoC validation, early software development or limited production testing. The HVP contains everything you need to launch your products in the shortest possible time including a binary FPGA-implementation of Arasan's market leading eMMC Host IP and software drivers running on a Linux OS which enables user written programs to fully utilize the controller functions.

The Arasan HVP contains a low profile 2.9 mm card connector applicable to D/SDIO/MMC/eMMC card types. Cards inserted into the connection can be accessed through Linux compatible user-written programs employing standard SD drivers. The system offers card auto detect and card bounce condition handling.

The SDIO/SD/eMMC Host Controller is fully compliant with the SD Host Controller version 3.0, SDIO card version 3.0, SD Memory Card Physical Layer version 3.0 specifications, and eMMC specifications version 5.0. Arasan's HVP supports both 3.3V and 1.8V signaling and handles the complete SDIO, SD/eSD, MMC/eMMC Protocol including transmission, data packing, CRC generation, start/end bit generation, and transaction format correctness checks.

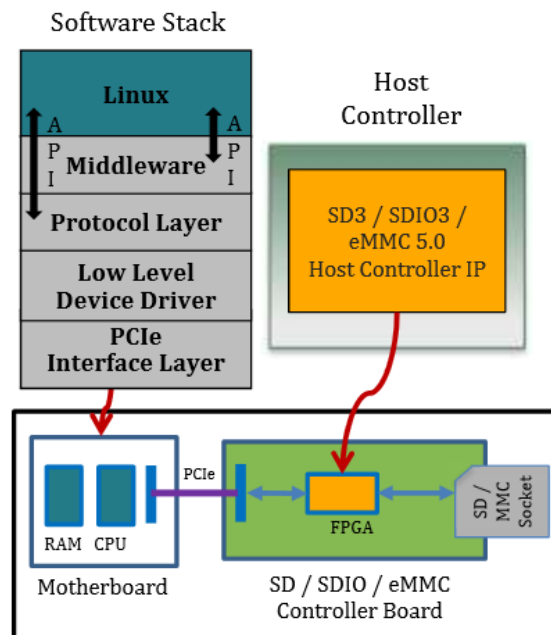


Figure 15: SD/SDIO/eMMC HVP Architecture

6.4 Deliverables

- SD/SDIO/eMMC Validation Platform including:
 - SD 3.0/SDIO 3.0/eMMC 5.0 Controller IP implemented in FPGA
 - Software Stack
- Linux Installer Package with Documentation
- Documentation

7 SD 3.0/SDIO 3.0/eMMC 4.51 Host Controller Software Stack

7.1 Overview

This is a production ready software stack for SD 3.0/ SDIO 3.0/ eMMC 4.51 Host Controller IP that is used to connect SD, SDIO or eMMC devices.

The SD 3.0/eMMC 4.51 stack can also be used for validating a device during its development and integration life cycles thereby helping designers to reduce the time to market for their product. The modular neX+ stack is architected to be OS and platform independent which eases porting effort. It has a thin OS and hardware abstraction layers making it highly portable. The neX+ stack provides a generic API set to access, control and configure the bus driver, host controller driver and the underlying hardware.

The SD 3.0/eMMC 4.51 software stack include functions for initialization, sending of commands, data transfer, power management, bus configuration, client driver matching, host controller hardware configuration and shutdown. The neX+ stack can support a single host controller with multiple slots or multiple host controllers with multiple slots. The software stack complies with the latest eMMC standards. It supports eMMC cards with the option for a device to boot directly from these cards using the boot mode feature.

7.2 Features

- Compliant with
 - SD Host Controller Specification v3.0
 - SDIO Specification v3.0
 - SD Memory Specification v3.0
 - eSD Specification v2.1
 - eMMC specification v4.5.1
 - MMC specification v4.3
 - Backward compatible to previous SD, eMMC, MMC specifications
- Auto detects UHS-I SD cards
- SD3.0 Point to Point and Multi device connection using Ring Connection topology
- Dual operating voltage range 3.3V and 1.8V
- High speed mode up to 208MHz SD host clock
- eMMC HS400 and HS200 operations
- eMMC Field firmware update
- eMMC device health report
- eMMC production state awareness
- eMMC Secure removal types

- eMMC Sleep notification
- Multiple host controllers support
- Multiple slots per host controller
- 1-bit, 4-bit SD modes
- 1-bit, 4-bit, and 8-bit MMC modes
- High Capacity Class 2, 4 and 6 SD cards
- Non DMA, Single operation DMA, and ADMA modes
- Byte and block mode transfers
- Card auto detection (Insertion / removal)
- Card bounce condition handling
- eSD and eMMC boot mode
- support API Interface
- Generic API interface abstracting protocol specific details

7.3 Architecture

The Arasan SD3.0/eMMC4.5.1 stack consists of the Application interface layer (API Layer), Protocol layer, Host controller driver layer, Hardware specific layer, and OS abstraction layer. The layered architecture allows porting to various operating systems and various platforms.

Client applications such as the block driver and SDIO function drivers interface with the API layer to use the device. The nitty-gritty details of the protocol is abstracted for the end-user and is all handled in the software stack. A set of well-defined APIs are provided at this layer. The SDIO client applications using different SDIO functions such as Bluetooth, Wi-Fi, UART, and etc. can register with the stack. The block driver integrates the flash devices such as SD, MMC memory cards to the Linux file system.

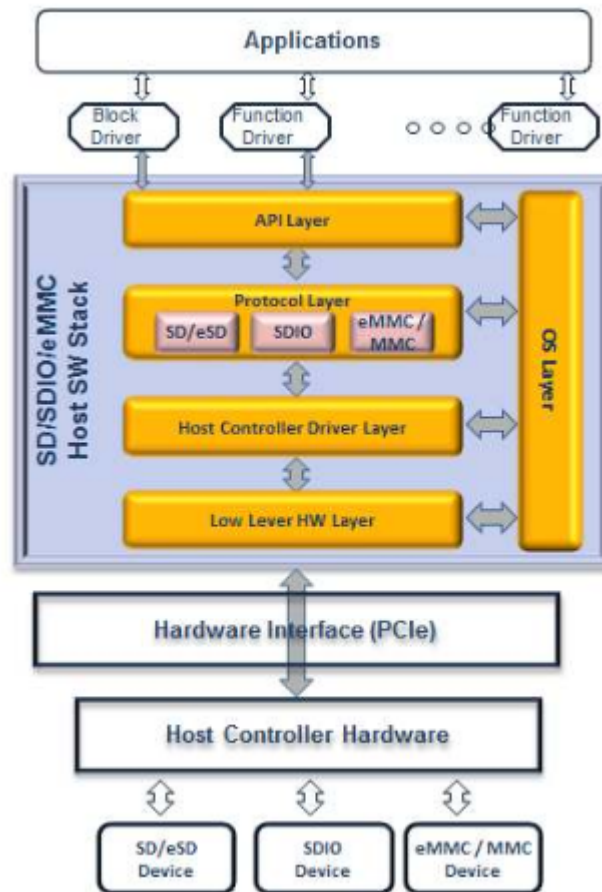


Figure 16: SD 3.0/eMMC 4.51 Stack Architecture

7.4 Deliverables

- Source code in C language
- API Guide
- User Manual

8 Services & Support

8.1 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.

8.2 Arasan Support Team

Our technical support is being provided 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.

8.3 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.

8.4 The Arasan Porting Engine

Analog IP blocks 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”.

8.5 Pricing & Licensing

Arasan is charging 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.