HB0104 Handbook CoreSDR_AHB v4.4

Microsemi

a Microchip company

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

50200104. 4.0 9/20

Contents

1 Revision History 1
- 1.1 Revision 4.0 1
- 1.2 Revision 3.0 1
- 1.3 Revision 2.0 1
- 1.4 Revision 1.0 1

2 Introduction 2
- 2.1 Core Overview 2
- 2.2 Core Version 2
- 2.3 Supported Device Families 3
- 2.4 Key Features 3
- 2.5 Device Utilization and Performance 4

3 Functional Block Description 5
- 3.1 Functional Overview 5
- 3.2 SDRAM Overview 6
- 3.3 CoreSDR_AHB Operation 8
  - 3.3.1 Initialization 8
- 3.4 Auto-Refresh 8
- 3.5 Bank Management 9

4 Tool Flows 10
- 4.1 License 10
  - 4.1.1 Register Transfer Level (RTL) 10
- 4.2 SmartDesign 10
- 4.3 Simulation Flows 11
- 4.4 Synthesis in Libero 11
- 4.5 Place-and-Route in Libero 11

5 Core Parameters 12
- 5.1 Parameters/Generics 12

6 Interface Descriptions 15
- 6.1 AHB Interface Signals 15
- 6.2 SDR SDRAM Interface Signals 16

7 Interface Timings 17
- 7.1 SDRAM Writes and Reads 17

8 Design Constraints 19

9 Testbench Operation and Modifications 21
- 9.1 Testbench Operation 21
  - 9.1.1 Verilog Testbench 21
- 9.2 Testbench Description 21

Figures

Figure 1 CoreSDR_AHB Application 2
Figure 2 CoreSDR_AHB Block Diagram 5
Figure 3 SDR Initialization Sequence 8
Figure 4 Refresh Timing 8
Figure 5 CoreSDR_AHB Configuration Within SmartDesign 10
Figure 6 CoreSDR_AHB Write and Read when write SYS_FREQ = 0 17
Figure 7 CoreSDR_AHB Write and Read when read SYS_FREQ = 1 17
Figure 8 CoreSDR_AHB Write and Read when write SYS_FREQ = 2 18
Figure 9 CoreSDR_AHB Burst Read 18
Figure 10 CoreSDR_AHB Write Burst Write 18
Figure 11 CoreSDR_AHB Testbench 21

Tables

Table 1 CoreSDR_AHB Device Utilization and Performance 4
Table 2 SDRAM Bus Commands 6
Table 3 Standard SDR SDRAM Device Configurations 7
Table 4 CoreSDR_AHB Generics 12
Table 5 Example Controller Parameter Values for CoreSDR_AHB 14
Table 6 Local Bus Signals 15
Table 7 SDR SDRAM Interface Signals 16

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 4.0

Updated the document for CoreSDR_AHB v4.4 release.

1.2 Revision 3.0

Updated the document for CoreSDR_AHB v4.3 release.

1.3 Revision 2.0

Updated the document for CoreSDR_AHB v4.1 release.

1.4 Revision 1.0

Updated the document for CoreSDR_AHB v4.0 release.

2 Introduction

2.1 Core Overview

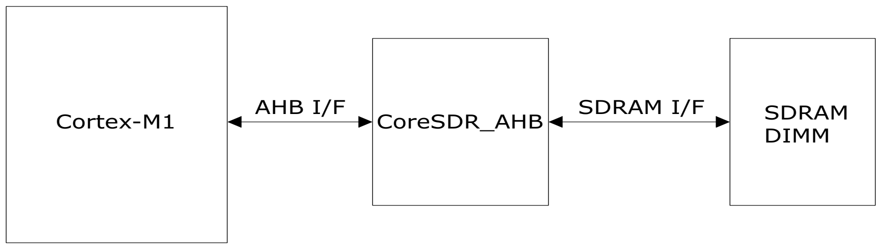
CoreSDR_AHB provides a high-performance interface to Single Data Rate (SDR) Synchronous Dynamic Random Access Memory (SDRAM) devices. CoreSDR_AHB accepts read and write commands using the Advanced High-performance Bus (AHB) slave interface and translates these requests to the command sequences required by SDRAM devices. The initialization and refresh functions are performed by CoreSDR_AHB.

CoreSDR_AHB uses bank management techniques to monitor the status of each SDRAM bank. Banks are only opened or closed when necessary, minimizing access delays. Up to four banks can be managed at a time. CoreSDR_AHB has the support of access cascading that allows read or write requests to be chained together. This results in no delay between requests, enabling up to 100% memory throughput for sequential accesses.

CoreSDR_AHB is provided with configurable memory settings (row bits and column bits) and timing parameters (CAS latency, t_{RAS}, t_{RC}, t_{RFC},t_{RCD}, t_{RP}, t_{MRD}, t_{RRD}, t_{REFC}, t_{WR}). The memory settings and timing parameters are configured through the configuration GUI. This ensures compatibility with virtually any SDRAM configuration.

A typical application using CoreSDR_AHB is shown in Figure 1.

Figure 1 • **CoreSDR_AHB Application**



2.2 Core Version

This handbook applies to CoreSDR_AHB v4.4. The release notes provided with the core, list the known discrepancies between this handbook and the core release associated with the release notes.

2.3 Supported Device Families

- PolarFire® SoC
- PolarFire®
- RTG4™
- IGLOO®2
- SmartFusion®2
- SmartFusion®
- ProASIC®3
- ProASIC3E
- ProASIC3L
- Fusion

2.4 Key Features

The following key features are supported in CoreSDR_AHB module:

- High performance, SDR controller for standard SDRAM chips and dual in-line memory modules (DIMMs)
- Synchronous interface, fully pipelined internal architecture
- Supports up to 1,024 MB of memory
- Bank management logic monitors status of up to 8 SDRAM banks
- Support for AHB slave interface
- Data access of 8, 16, or 32 bits are allowed by masters

2.5 Device Utilization and Performance

CoreSDR_AHB has been implemented in several Microsemi® device families. A summary of the implementation data is listed in Table 1.

Table 1 • **CoreSDR_AHB Device Utilization and Performance**

FPGA Family and Device	Parameter	Tiles			Utilization	Frequency (Mhz)	
	BURST_SUPPORT	Sequential	Comb	Total		HCLK	SDRCLK_IN
PolarFire SoC MPFS250T_ES	0	686	767	1453	0.285	194	259.5
	1	878	3071	3949	0.78	130	255
PolarFire MPF500T-1FCG1152E	0	684	728	1412	0.145	228.8	312.1
	1	878	3067	3945	0.41	153	260
RTG4 RT4G150-1FCG1657M	0	691	801	1492	0.49	183	199
	1	957	3225	4182	0.87	110	137
IGLOO2 M2GL050T-FG484	0	685	742	1427	2.54	160.2	195.5
	1	956	3185	4150	3.68	115	213
SmartFusion2 M2S050T- FG484	0	686	797	1483	1.315	160.2	195.5
	1	965	3185	4150	3.68	115	213
ProASIC3 A3P600-2	0	669	998	1667	12	91	124
ProASIC3E A3PE600-2		669	998	1667	12	91	124
ProASIC3L A3PE600L-1FG484M		672	1015	1687	12	75	116
ProASIC3 A3P600-2	1	4690	4690	5550	40	66	108
ProASIC3E A3PE600-2FG484		860	4690	5550	40	66	108
ProASIC3L A3PE600L-1FG484M		862	4684	5546	40	60	95
Fusion AFS600-2FG484	0	669	998	1667	12	91	124
	1	860	4690	5550	40	66	108
SmartFusion A2F500M3G-FG484	0	672	1117	1789	16	92	100
	1	874	4756	5630	48	50	79

Note: All data was obtained using a default system configuration. All performance data was obtained under commercial (COM) conditions.

3 Functional Block Description

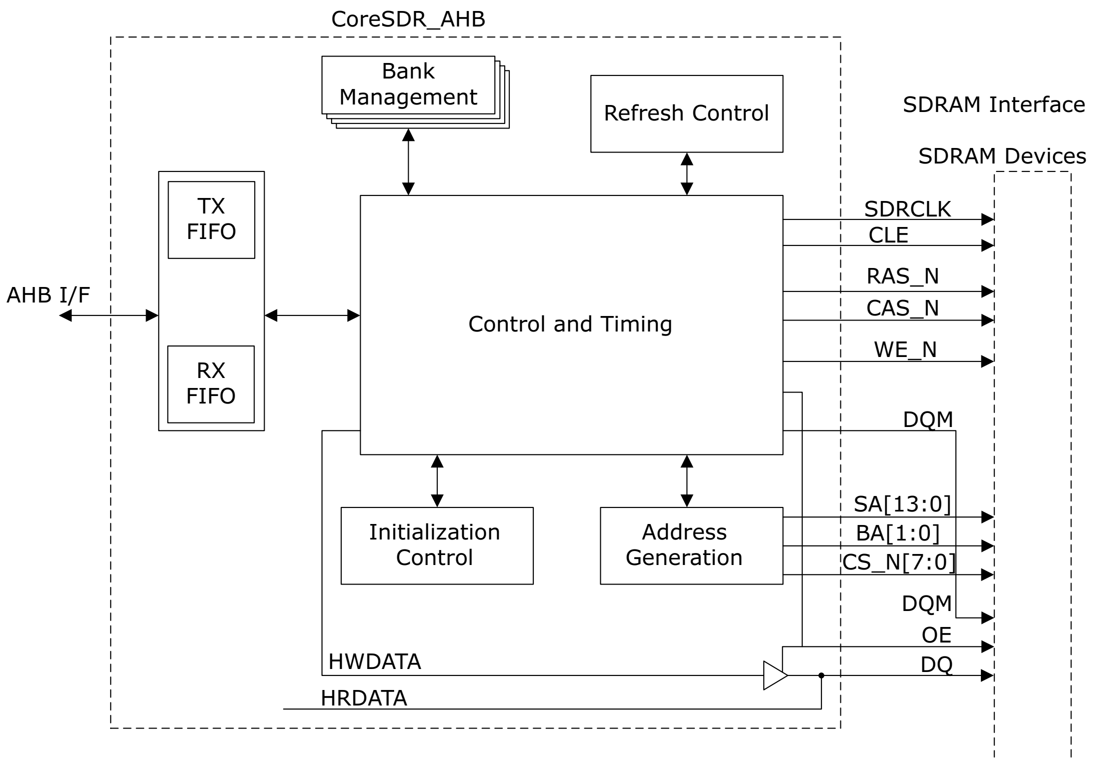
3.1 Functional Overview

CoreSDR_AHB consists of the following primary blocks, as shown in Figure 2:

- AHB Lite interface wrapper -
  - When BURST_SUPPORT=1, the AHB wrapper instantiates a two asynchronous FIFO to handle TX and RX data path between AHB HCLK and SDRCLK_IN clock domain to support AHB Burst transactions
  - When BURST_SUPPORT=0, no asynchronous FIFO implemented
- Control and Timing Block - Main controller logic
- Initialization Control - Performs initialization sequence after RESET_N is deactivated or SD_INIT is pulsed
- Address Generation - Puts out an address, bank address, and chip select signals on the SDRAM interface
- Bank Management - Keeps track of last opened row and bank to minimize command overhead
- Refresh Control - Performs automatic refresh commands to maintain data integrity

The tristate buffer shown in Figure 2 resides in the I/O, and its output enable is controlled by the core.

Figure 2 • **CoreSDR_AHB Block Diagram**



3.2 SDRAM Overview

The synchronous interface and fully pipelined internal architecture of SDRAM allow extremely fast data rates if used efficiently. SDRAM is organized in banks of memory addressed by row and column. The number of row and column address bits depends on the size and configuration of the memory.

SDRAM is controlled by bus commands that are formed using combinations of the RAS_N, CAS_N, and WE_N signals. For instance, on a clock cycle where all three signals are HIGH, the associated command is a no operation (NOP). A NOP is also indicated when the chip select is not asserted. The standard SDRAM bus commands are listed in Table 2.

Table 2 • **SDRAM Bus Commands**

Command	RAS_N	CAS_N	WE_N
NOP	H	H	H
Active	L	H	H
Read	H	L	H
Write	H	L	L
Burst Terminate	H	H	L
Precharge	L	H	L
Auto-Refresh	L	L	H
Load Mode Register	L	L	L

SDRAM devices are typically divided into four banks. These banks must be opened before a range of addresses can be written to or read from. The row and bank to be opened are registered coincident with the active command. When a new row on a bank is accessed for a read or a write, it is necessary to first close the bank and then reopen it to the new row. The precharge command closes a bank. Opening and closing banks cost memory bandwidth, so CoreSDR_AHB has been designed to monitor and manage the status of the four banks simultaneously. This enables the controller to intelligently open and close banks only when necessary.

When the read or write command is issued; the initial column address is presented to the SDRAM devices. In the case of SDR SDRAM, the initial data is presented concurrent with the write command. For the read command, the initial data appears on the data bus 2-3 clock cycles later. This is known as CAS latency and is due to the time required to physically read the internal DRAM and register the data on the bus. The CAS latency depends on the speed grade of the SDRAM and the frequency of the memory clock. In general, the faster the clock, the more cycles of CAS latency are required. After the initial read or write command, sequential reads and writes continues until the burst length is reached or a burst terminate command is issued. SDRAM devices support a burst length of up to eight data cycles.

CoreSDR_AHB is capable of cascading bursts to maximize SDRAM bandwidth. SDRAM devices require periodic refresh operations to maintain the integrity of the stored data. CoreSDR_AHB automatically issues the auto-refresh command periodically. No user intervention is required.

The load mode register command is used to configure the SDRAM operation. This register stores the CAS latency, burst length, burst type, and write burst mode. CoreSDR_AHB supports a sequential burst type and programmed-length write burst mode. The SDR controller only writes to the base mode register. Consult the SDRAM device specification for additional details on these registers.

In SDRAM, each bank is an organized block of rows and columns. A data width of 4, 8, or 16 bits is written to or read from the SDR SDRAM by providing the bank, row, and column addresses. To reduce pin count, SDRAM row and column addresses are multiplexed on the same pins. Table 3 lists the number of rows, columns, banks, and chip selects required for various standard discrete SDR SDRAM devices.

The x4, x8, and x16 stand for data width. The user needs to check the SDRAM specification and set the row and column bits. CoreSDR_AHB supports any of these devices.

Table 3 • **Standard SDR SDRAM Device Configurations**

Chip Size	Configuration	Rows	Columns	Banks
64 Mb	16M×4	12	10	4
64 Mb	8M×8	12	9	4
64 Mb	4M×16	12	8	4
64 Mb	2M×32	11	8	4
128 Mb	32M×4	12	11	4
128 Mb	16M×8	12	10	4
128 Mb	8M×16	12	9	4
128 Mb	4M×32	12	8	4
256 Mb	64M×4	13	11	4
256 Mb	32M×8	13	10	4
256 Mb	16M×16	13	9	4
512 Mb	128M×4	13	12	4
512 Mb	64M×8	13	11	4
512 Mb	32M×16	13	10	4
1,024 Mb	256M×4	14	12	4
1,024 Mb	128M×8	14	11	4
1,024 Mb	64M×16	14	10	4

SDRAM is typically available in DIMMs, small outline DIMMs (SO-DIMMs), and discrete chips. The number of row and column bits for a DIMM or SO-DIMM configuration can be found by determining the configuration of the discrete chips used on the module. This information is available in the module datasheet.

3.3 CoreSDR_AHB Operation

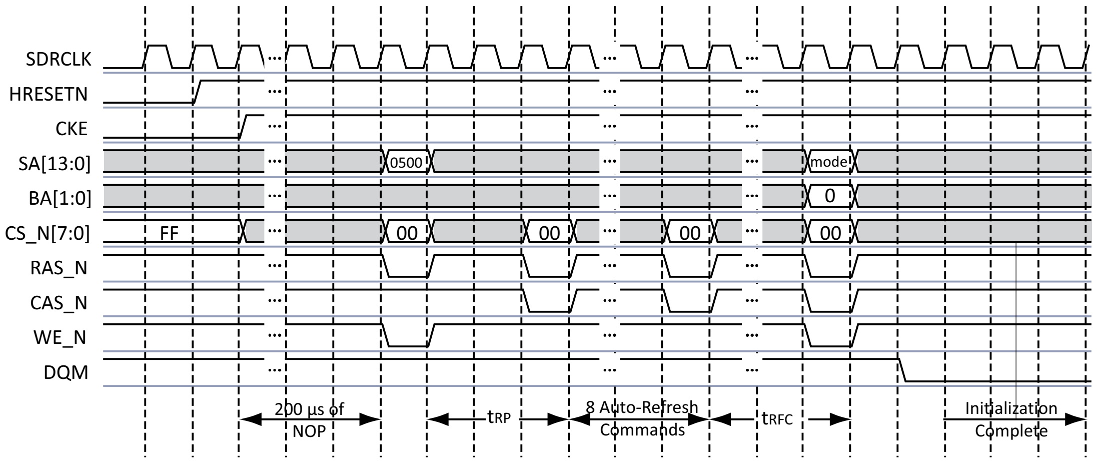
3.3.1 Initialization

After SDRCLK_RESETN is deasserted, CoreSDR_AHB performs the following sequence:

1. NOP command is issued for 200 µs (period controlled by the delay port parameter)
2. Precharge-all command
3. Eight auto-refresh commands
4. Load mode register command - This causes the SDRAM mode register to be loaded with the proper burst length (BL) and CAS latency (CL) values.

CoreSDR_AHB initialization timing is shown in Figure 3.

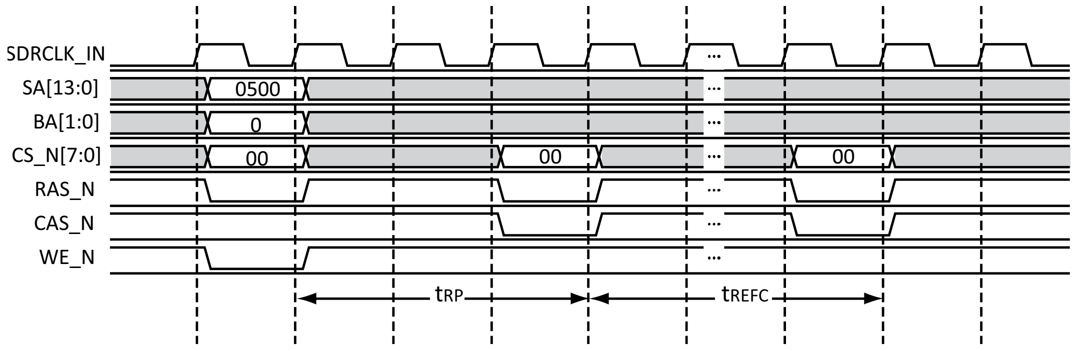
Figure 3 • **SDR Initialization Sequence**



3.4 Auto-Refresh

SDRAM devices require periodic auto-refresh commands to maintain data integrity. CoreSDR_AHB automatically issues periodic auto-refresh commands to the SDRAM device(s) without user intervention. The refresh period configuration port (ref) specifies the period between refreshes, in clock cycles. Figure 4 shows an example of two refresh commands. The first refresh sequence occurs when one or more banks have been left open as a result of a read without precharge or write without precharge operation. All open banks are closed using the precharge-all command (RAS_N, WE_N asserted with sa[10] and sa[8]) before the refresh command. In Figure 4, a refresh occurs again after the refresh period has elapsed, as determined using the ref configuration port. The refresh never interrupts a read or write in the middle of a data burst. However, if the controller determines that the refresh period has elapsed at a point concurrent with or before a read or write request, the request may be held off (RW_ACK will not get asserted) until after the refresh has been performed.

Figure 4 • **Refresh Timing**



3.5 Bank Management

CoreSDR_AHB incorporates bank management techniques to minimize command overhead. For each bank, the controller records the last opened row and whether or not the bank has been closed. When a local bus interface read or write request occurs, CoreSDR_AHB checks to determine if the requested bank is already opened and whether the request is for the same row as the one the bank is already opened with. If the bank is already opened with the requested row, CoreSDR_AHB performs the function immediately. If the bank is opened to a different row, the controller closes the bank (using the precharge command) and reopens the bank (using the active command) to the requested row. If the bank is already closed, the controller opens the bank to the requested row (using the active command).

Requests to the controller can be issued as read with auto-precharge, write with auto-precharge, read without auto-precharge, and write without auto-precharge. Commands are issued with auto-precharge if the AUTO_PCH parameter is set concurrent with the read or write request (W_REQ) signals. After a read with auto-precharge or write with auto-precharge, the accessed bank is automatically closed internally by the SDRAM device(s). After a read without auto- precharge or write without auto-precharge, the accessed bank is left open until closing is required. Closing will occur whenever a request is issued to a row different from the row a bank is already open to, or during the next refresh sequence. The refresh sequence will close all the banks (using the precharge-all command) if all banks are not already closed.

The default configuration of the controller tracks the status of four banks at a time. This means that an access to row **a** on bank **a** on chip select **a** is treated differently from an access to row **a** on bank **a** on chip select **b**. Therefore, a close and open sequence is performed when switching between these rows.

4 Tool Flows

4.1 License

No license is required for this core.

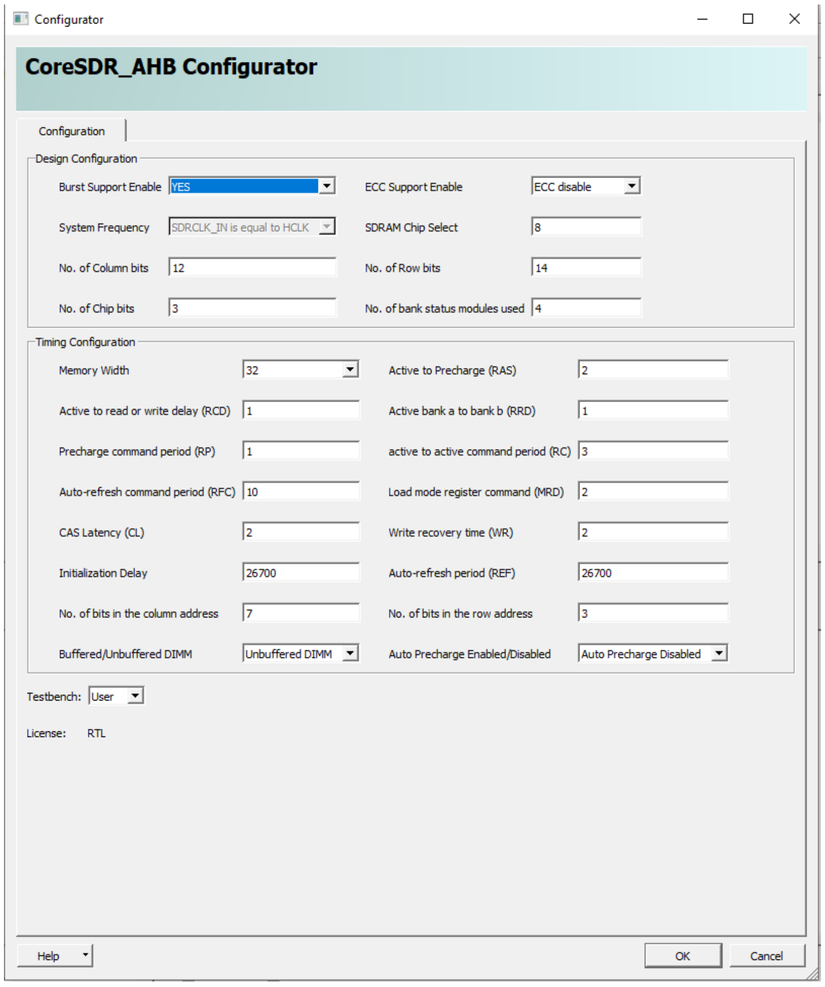
4.1.1 Register Transfer Level (RTL)

Complete RTL source code is provided for the core and testbenches.

4.2 SmartDesign

CoreSDR_AHB is available for download in the Libero SoC IP catalog through the web repository. Once it is listed in the catalog, the core can be instantiated using the SmartDesign flow. For more information about using SmartDesign to configure, connect, and generate cores, see *Using DirectCore in Libero User Guide* or consult the *Libero SoC online help*. The core can be configured using the configuration GUI within SmartDesign, as shown in Figure 5.

Figure 5 • **CoreSDR_AHB Configuration Within SmartDesign**



CoreSDR_AHB Configurator

Design Configuration

Burst Support Enable	YES	ECC Support Enable	ECC disable
System Frequency	SDRCLK_IN is equal to HCLK	SDRAM Chip Select	8
No. of Column bits	12	No. of Row bits	14
No. of Chip bits	3	No. of bank status modules used	4

Timing Configuration

Memory Width	32	Active to Precharge (RAS)	2
Active to read or write delay (RCD)	1	Active bank a to bank b (RRD)	1
Precharge command period (RP)	1	active to active command period (RC)	3
Auto-refresh command period (RFC)	10	Load mode register command (MRD)	2
CAS Latency (CL)	2	Write recovery time (WR)	2
Initialization Delay	26700	Auto-refresh period (REF)	26700
No. of bits in the column address	7	No. of bits in the row address	3
Buffered/Unbuffered DIMM	Unbuffered DIMM	Auto Precharge Enabled/Disabled	Auto Precharge Disabled

Testbench: User

License: RTL

4.3 Simulation Flows

To run simulations, select the user testbench within the SmartDesign CoreSDR_AHB configuration GUI, right-click, and select **Generate Design** (Figure 5).

When SmartDesign generates the design files, it will install the appropriate testbench files. To run the simulation, set the design root to the CoreSDR_AHB instantiation in the Libero design hierarchy pane, and click **Simulation** in the Libero **Design Flow** window. This invokes QuestaSim® and automatically runs the simulation.

4.4 Synthesis in Libero

Set the design root appropriately and click the Synthesis icon in the Libero. The synthesis window appears, displaying the Synplicity® project. Set Synplicity to use the Verilog 2001 standard if Verilog is being used. To perform synthesis, click **Run**.

4.5 Place-and-Route in Libero

After the design has been synthesized, click **Layout** in Libero to invoke Designer. CoreSDR_AHB requires no special place-and-route settings.

5 Core Parameters

5.1 Parameters/Generics

The generics are listed in Table 4 as required in the source code.

Table 4 • **CoreSDR_AHB Generics**

Generic	Default Setting	Valid Values	Description
BURST_SUPPORT	1	0 and 1	0 - NO, AHB BURST support Disabled 1 - YES, AHB BURST support Enabled
SDRAM_CHIPS	8	1 to 8	Number of chip selects
SDRAM_COLBITS	12	8 to 12	Maximum number of SDRAM column bits
SDRAM_ROWBITS	14	11 to 14	Maximum number of SDRAM row bits
SDRAM_CHIPBITS	3	1 to 3	Number of encoded chip select bits
SDRAM_BANKSTAT MODULES	4	4 and 8	Number of bank status modules used (For more information, see Bank Management, page 9.)
SYS_FREQ	1	0, 1, and 2	SYS_FREQ parameter is enabled when BURST_SUPPORT parameter is 0 to support only AHB single transaction. When SYS_FREQ is 0, the HCLK will be directly connected to the SDRCLK_IN. When SYS_FREQ is 1, the SDRCLK_IN frequency will be 2 × HCLK (where HCLK is 1 to 66 MHz). When SYS_FREQ is 2, the SDRCLK frequency will be 4 × HCLK (where HCLK is 1 to 33 MHz).
ECC	0	0 and 1	0 - ECC Disabled 1 - ECC Enabled
DQ_SIZE	32	16, 32	Width of SDRAM data bus (DQ). This determines the byte mapping of AHB to SDR.
FAMILY	16	15 16 17 18 19 22 24 25 26 27	Must be set to match the supported FPGA family. 15 - ProASIC3 16 - ProASIC3E 17 - Fusion 18 - SmartFusion® 19 - SmartFusion®2 22 - ProASIC®3L 24 - IGLOO®2 25 - RTG4 26 - PolarFire® 27 - PolarFire® Soc
RAS	6	1-10	SDRAM active to precharge (t_{RAS}), specified in clock cycles
RCD	3	2-5	SDRAM active to read or write delay (t_{RCD}), specified in clock cycles
RRD	2	2-3	SDRAM active bank **a** to active bank **b** (t_{RRD}), specified in clock cycles
RP	3	1-4	SDRAM precharge command period (t_{RP}), specified in clock cycles
RC	8	3-12	SDRAM active to active/auto-refresh command period (t_{RC}), specified in clock cycles

Table 4 • **CoreSDR_AHB Generics *(continued)***

Generic	Default Setting	Valid Values	Description
RFC	9	2-14	Auto-refresh to active/auto-refresh command period (t_{RFC}), specified in clock cycles
MRD	2	1-7	SDRAM load mode register command to active or refresh command (t_{MRD}), specified in clock cycles
CL	2	1-4	SDRAM CAS latency, specified in clock cycles
WR	2	1-3	SDRAM write recovery time (t_{WR})
DELAY	26,700	10-65,535	Delay after a reset event that the controller waits before initializing the SDRAM, specified in clock cycles. Per JEDEC standards, SDR devices require this delay to be a minimum of 200 µs.
REF	26,700	10-65,535	Period between auto-refresh commands issued by the controller, specified in clock cycles. REF = auto refresh interval / tCK, where tCK is the clock cycle in ns.
COLBITS	4	3-7	Number of bits in the column address (encoded). Values are decoded as follows: 3: 8 column bits 4: 9 column bits 5: 10 column bits 6: 11 column bits 7: 12 column bits
ROWBITS	2	0-3	Number of bits in the row address (encoded). Values are decoded as follows: 0: 11 row bits 1: 12 row bits 2: 13 row bits 3: 14 row bits
REGDIMM	0	0-1	Set when using registered/buffered DIMMs. Causes adjustment in local bus interface timing to synchronize with SDRAM command timing delayed by register/buffer on DIMM.
AUTO_PCH	0	0-1	When asserted in conjunction with R_REQ or W_REQ, causes command to be issued as read with auto-precharge or write with auto-precharge, respectively.

For example:

If RCD is 20 ns in the specification and the clock is 133 MHz, then the number of clocks equals to 20/7.5=3. The user needs to hardcode the RCD ports to the binary value 011 at the top level since the value is 3.

If RCD is 20 ns in the specification and the clock is 100 MHz, then the number of clocks equals to 20/10=2. If it is divisible without a remainder, then set it to the next integer, which is 3 in the previous example, so that there will be little margin. The user needs to set the parameter value to 3.

Example settings for the timing-related parameters are shown in Table 5. These settings are based on the speed grade of the SDRAM devices and the desired operating frequency. For more information about the specific timing values of the SDRAM device, consult the datasheet.

Table 5 • **Example Controller Parameter Values for CoreSDR_AHB**

	100 MHz (10 ns period)[1]		133 MHz (7.518 ns period)[2]	
Parameter	**Specification**	**Value**	**Specification**	**Value**
RAS	44.0 ns	5	37.0 ns	5
RCD	20.0 ns	3	15.0 ns	2
RRD	15.0 ns	2	14.0 ns	2
RP	20.0 ns	3	15.0 ns	2
RC	66.0 ns	7	60.0 ns	8
RFC	66.0 ns	7	66.0 ns	9
MRD	2 clks	2	2 clks	2
CL	-	2	-	2
WR	15.0 ns	2	14.0 ns	2
DELAY	100 µs	10000	100 µs	13302
REF	7.8125 µs	781	7.8125 µs	1040

1. Values based on Micron MT48LC32M8A2-75
2. Values based on Micron MT48LC32M8A2-7E

6 Interface Descriptions

The port signals for CoreSDR_AHB are defined in Table 6 and Table 7. The port signals are also shown in Figure 2, page 5. All signals are designated either input (input-only) or output (output-only), except DQ, which is bidirectional.

6.1 AHB Interface Signals

The user interface to CoreSDR_AHB is referred to as the local bus interface. The local bus signals are listed in Table 6.

Table 6 • **Local Bus Signals**

Signal	I/O	Description
SDRCLK_IN	Input	SDR Core Clock
SDRCLK_RESETN	Input	SDR Core reset (Active low and asynchronous)
HCLK	Input	AHB clock
HRESETN	Input	AHB reset (Active low and asynchronous)
HADDR[31:0]	Input	AHB address
HREADYIN	Input	AHB ready in
HTRANS[1:0]	Input	AHB transfer type
HWRITE	Input	AHB write/read
HSIZE[2:0]	Input	AHB transfer size
HBURST[2:0]	Input	AHB Burst Type
HSEL	Input	AHB slave select
HREADY	Output	AHB ready out
HRESP[1:0]	Output	AHB response
HWDATA[31:0]	Input	AHB data in
HRDATA[31:0]	Output	AHB data out

6.2 SDR SDRAM Interface Signals

The external interface to SDRAM devices is referred to as the SDRAM interface. The SDRAM interface signals are listed in Table 7.

Table 7 • **SDR SDRAM Interface Signals**

Signal	Name	I/O	Description
SDRCLK_OUT	SDRAM Clock	Output	SDRAM clock drives the SDRAM DIMMS, frequency is same as SDRCLK_IN input clock frequency.
SA[13:0]	Address Bus	Output	Sampled during the active, precharge, read, and write commands. This bus also provides the mode register value during the load mode register command.
BA[1:0]	Bank Address	Output	Sampled during active, precharge, read, and write commands to determine which bank command is to be applied to.
CS_N[7:0]	Chip Selects	Output	SDRAM chip selects
CKE	Clock Enable	Output	SDRAM clock enable. Held LOW during reset to ensure SDRAM DQ and DQS outputs are in the high-impedance state.
RAS_N	Row Address Strobe	Output	SDRAM command input
CAS_N	Column Address Strobe	Output	SDRAM command input
WE_N	Write Enable	Output	SDRAM command input
DQM	Data Mask	Output	SDRAM data mask asserted by controller during SDRAM initialization and during burst terminate. User may sum with user data mask bits.
OE	Output Enable	Output	Tristate control for DQ data
DQ	SDRAM in/out data	Inout	When OE=1, it drives read data, otherwise it drives write data.

7 Interface Timings

7.1 SDRAM Writes and Reads

Figure 6 shows an example CoreSDR_AHB write when SYS_FREQ = 0.

Figure 6 • **CoreSDR_AHB Write and Read when write SYS_FREQ = 0**

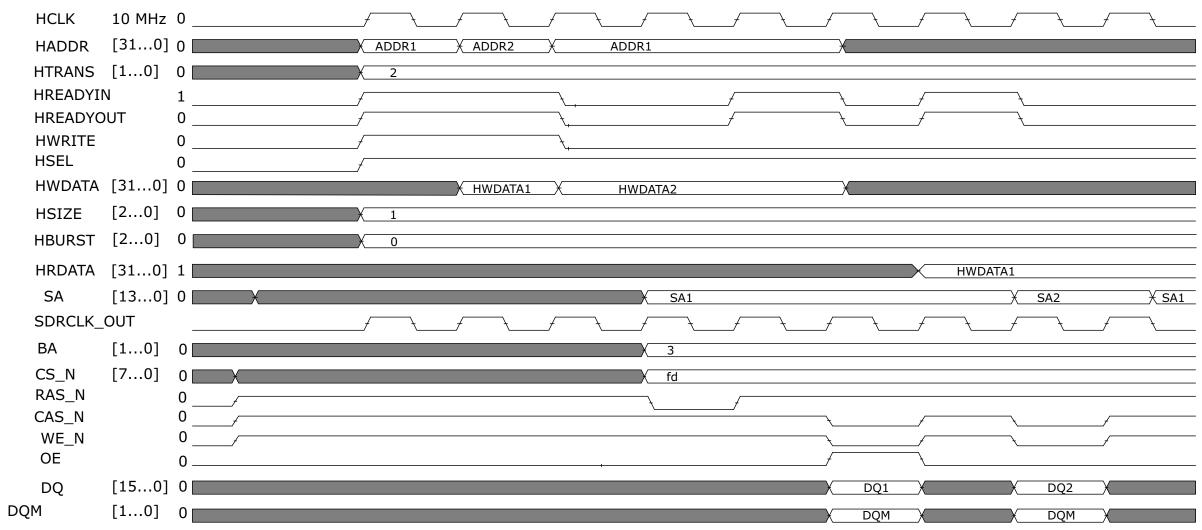


Figure 7 shows an example CoreSDR_AHB read when SYS_FREQ = 1.

Figure 7 • **CoreSDR_AHB Write and Read when read SYS_FREQ = 1**

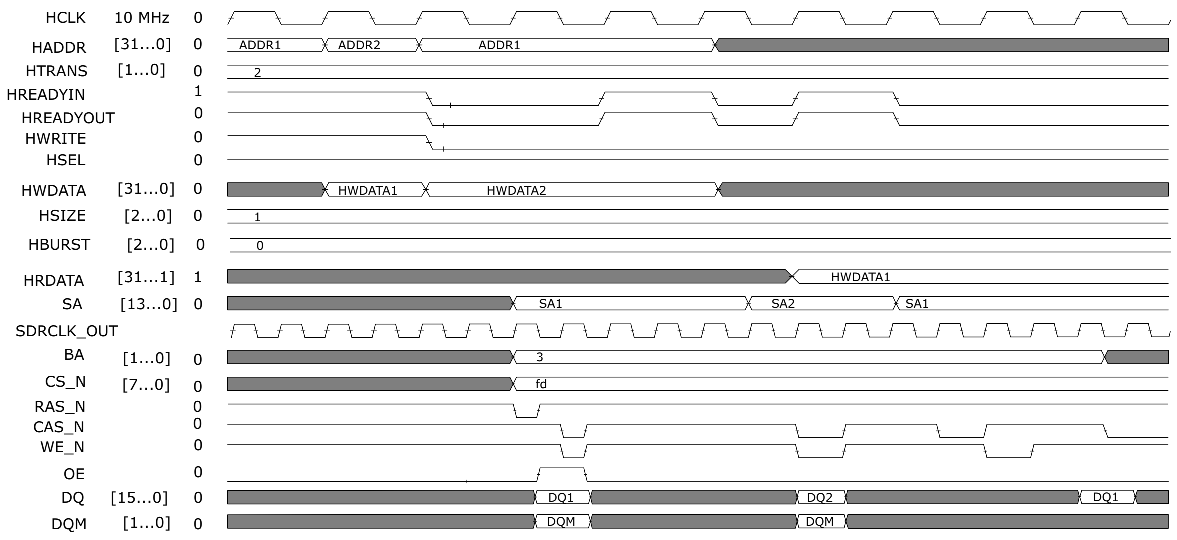


Figure 8 shows an example CoreSDR_AHB write when SYS_FREQ = 2.

Figure 8 • **CoreSDR_AHB Write and Read when write SYS_FREQ = 2**

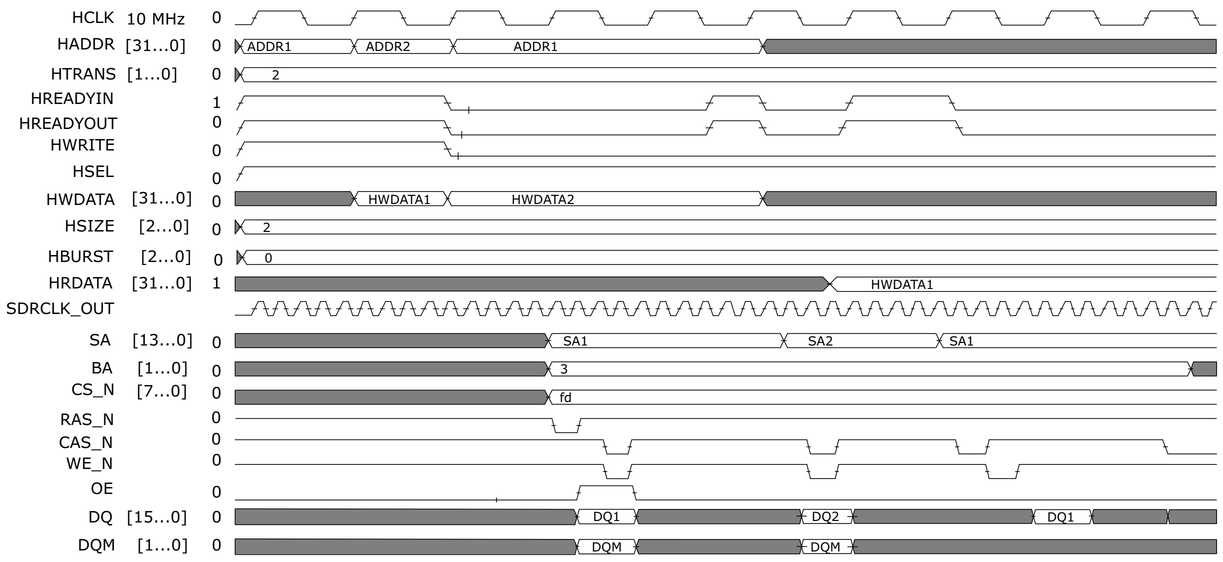


Figure 9 • **CoreSDR_AHB Burst Read**

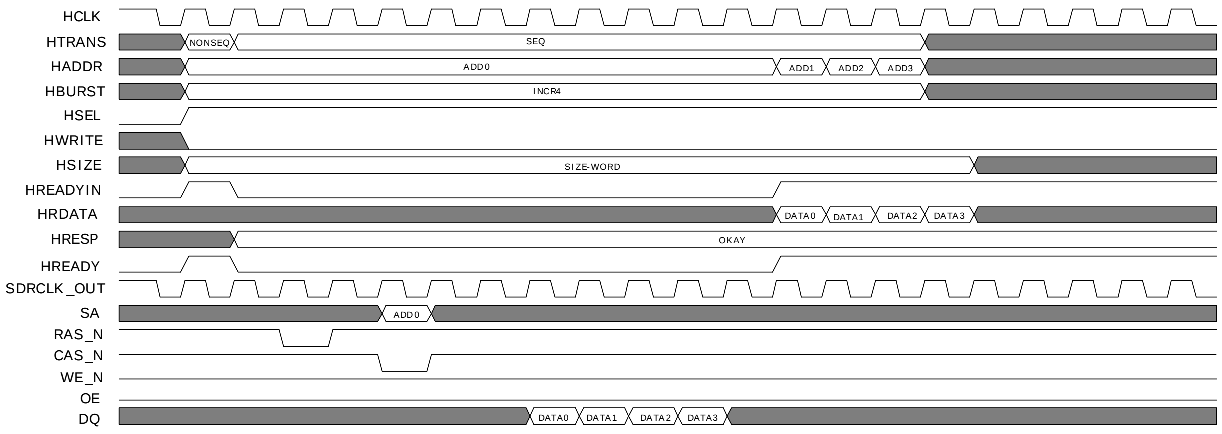
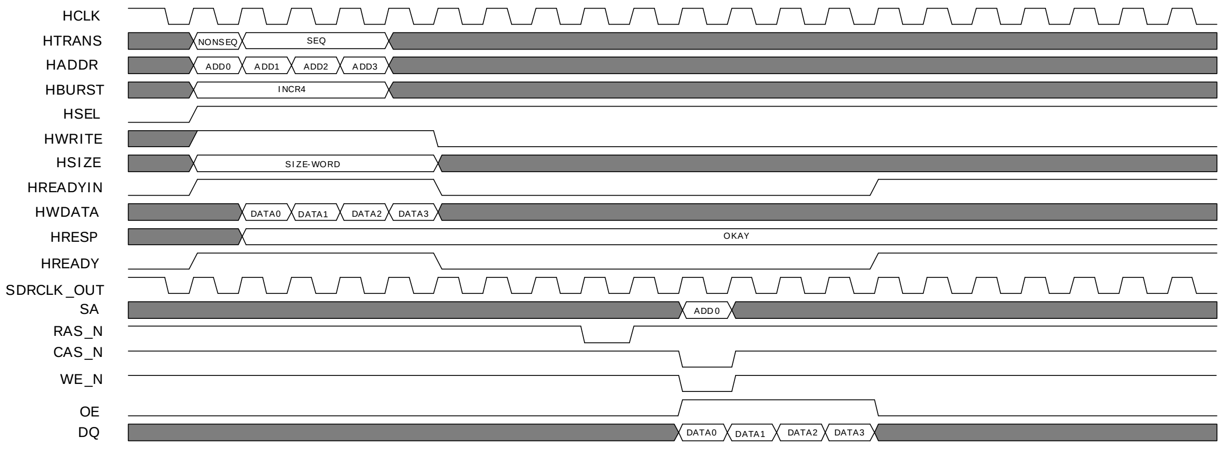


Figure 10 • **CoreSDR_AHB Write Burst Write**



8 Design Constraints

Designs that contain CoreSDR_AHB require the following constraints that are to be applied in the design flow to prevent timing violations occurring in the design.

Add the constraints for the path from HCLK clock domain to SDRCLK_IN clock domain and from SDRCLK_IN clock domain to HCLK domain is as follows:

CORESDR_AHB_HIER_INST --- Instance name of CoreSDR_AHB module.

When parameter BURST_SUPPORT =1 and DQ_SIZE = 32

```
set_false_path -from [get_cells\
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/tx_data_fifo_async/wrGrayCounter/cn
tGray*}]\

-to [get_cells
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/tx_data_fifo_async/wrPtr_s1*}]

set_false_path -from [get_cells\
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/rx_data_fifo_async/wrGrayCounter/cn
tGray*}]\

-to [get_cells
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/rx_data_fifo_async/wrPtr_s1*}]

set_false_path -from [get_cells\
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/wvalid_ahb/genblk1.pulse_gen_i/togg
le_out}]\

-to [get_cells\
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/wvalid_ahb/genblk2.pulse_cdc_sync_i
/sync_ff[0]}]

set_false_path -from [get_cells\
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/rvalid_ahb/genblk1.pulse_gen_i/togg
le_out}]\

 -to [get_cells\
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/rvalid_ahb/genblk2.pulse_cdc_sync_i
/sync_ff[0]}]

set_false_path -from [get_cells\
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/burst_terminate_ahb}]\

-to [get_cells
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/burst_terminate_sdr_s}]

set_false_path -from [get_cells
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/state_ahb*}]\

 -to [get_cells {CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/state_sdr*}]

set_false_path -from [get_cells
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/raddr_ahb*}]\

 -to [get_cells {CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/raddr_sdr_d*}]

set_false_path -from [get_cells
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/B_SIZE_ahb*}]\

-to [get_cells {CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/B_SIZE_sdr_d*}]
```

When parameter BURST_SUPPORT =1 and DQ_SIZE = 16

```
set_false_path -from [get_cells\
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/tx_data_fifo_async/wrGrayCounter/cn
tGray*}]\

-to [get_cells
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/tx_data_fifo_async/wrPtr_s1*}]

set_false_path -from [get_cells\
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/rx_data_fifo_async/wrGrayCounter/cn
tGray*}]\

-to [get_cells
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/rx_data_fifo_async/wrPtr_s1*}]

set_false_path -from [get_cells\
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/wvalid_ahb/genblk1.pulse_gen_i/togg
le_out}]\

-to [get_cells\
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/wvalid_ahb/genblk2.pulse_cdc_sync_i
/sync_ff[0]}]

set_false_path -from [get_cells\
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/rvalid_ahb/genblk1.pulse_gen_i/togg
le_out}]\

 -to [get_cells\
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/rvalid_ahb/genblk2.pulse_cdc_sync_i
/sync_ff[0]}]

set_false_path -from [get_cells\
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/burst_terminate_ahb}]\

-to [get_cells
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/burst_terminate_sdr_s}]

set_false_path -from [get_cells
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/state_ahb*}]\

 -to [ get_cells  { CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/state_sdr*}]

set_false_path -from [get_cells
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/raddr_ahb*}]\

 -to [get_cells {CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/raddr_sdr_d*}]

set_false_path -from [get_cells
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/B_SIZE_ahb*}]\

-to [get_cells {CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/B_SIZE_sdr_d*}]

set_false_path -from [get_cells
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/command*}] -to [get_cells
{CORESDR_AHB_HIER_INST/genblk1.CoreSDR_0/command_size_sdr_s*}]
```

9 Testbench Operation and Modifications

9.1 Testbench Operation

Following testbench is provided with CoreSDR_AHB:

- Verilog testbench

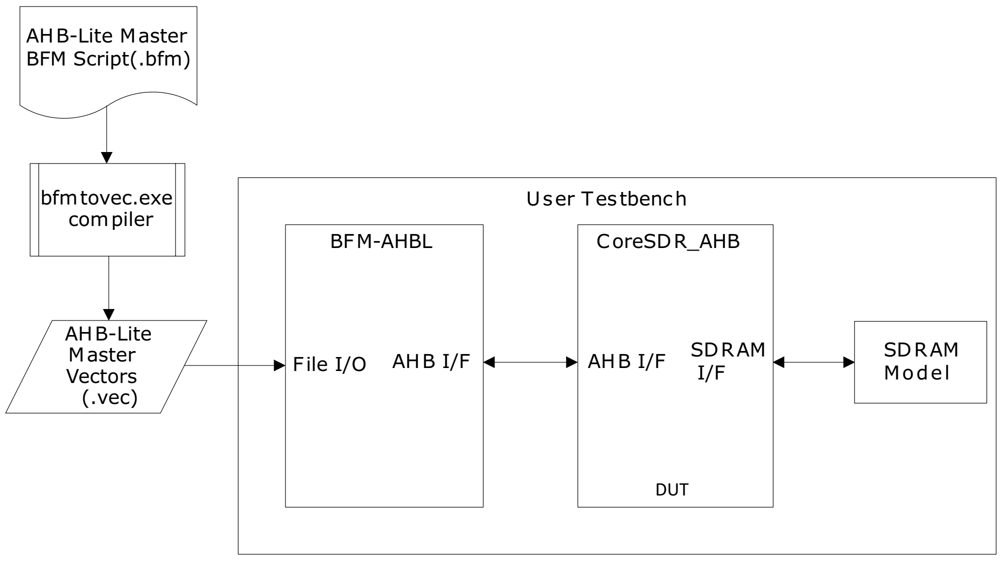
9.1.1 Verilog Testbench

The Verilog user testbench is provided as a reference and can be modified to suit the requirements. The source code for the Verilog user testbench is provided to ease the process of integrating the CoreSDR_AHB macro into the design and verifying its functionality.

9.2 Testbench Description

A user testbench is included with the RTL release of CoreSDR_AHB. A simplified block diagram of the testbench is shown in Figure 11. By default, the Verilog version, tb_user.v, instantiates a Micron 256 Mbit SDRAM model (MT48LC16M16A2.v, 4M×16 × 4 banks). The testbench instantiates the Design Under Test (DUT), which is the CoreSDR_AHB, the SDRAM model, as well as the test vector modules that provide stimuli sources for the DUT. A procedural testbench controls each module and applies the sequential stimuli to the DUT.

Figure 11 • **CoreSDR_AHB Testbench**



A commented Bus Functional Model (BFM) ASCII script source file (`.bfm`) is included in the following directory: YourLiberoProjectDirectory/ simulation, where the Libero Project Dir represents the path to the Libero SoC project where CoreSDR_AHB is used. The BFM source file is for controlling the AHB-Lite master and is named master.bfm. The BFM source file is automatically recompiled each time the simulation is invoked from Libero SoC by bfmtovec.exe if running on a Windows® platform, or by bfmtovec.lin if running on a Linux platform. The output `.vec` file created by the bfmtovec executable is read in by the BFM modules for simulation in QuestaSim®.

The BFM scripts can be altered, if desired. For more information, see *DirectCore AMBA BFM User Guide*. The source code for the user testbench, BFM scripts, and compiled QuestaSim simulation library containing the BFM modules are available with the CoreSDR_AHB RTL release.