SD Memory Slave Controller IP Integration Manual

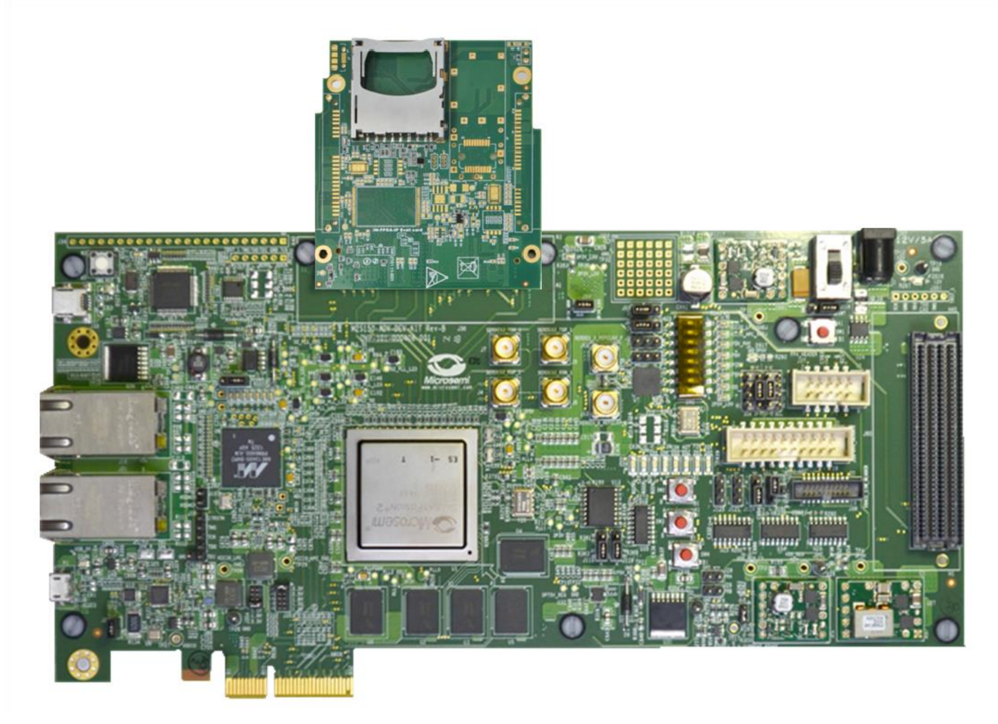


Table Of Contents

List Of Figures

List Of Tables

1 Introduction

1.1 Purpose

The purpose of this document is to describe SD Slave Controller IP Integration details and Testing Procedure on the SmartFusion2 Development Kit.

1.2 Reference Document

- SD Physical layer specification version 2.0

1.3 Overview

SD Memory Slave Controller will send the response to the SD host depending on the command issued.

1.4 Acronyms and Abbreviations

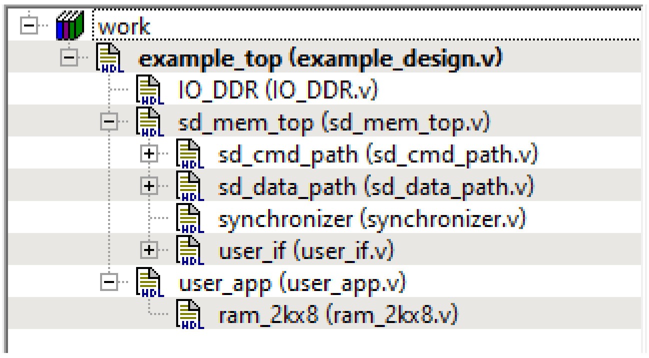
Table 1: Acronyms & Abbreviations

Term	Meaning
BRAM	Block RAM
FF	Flip Flop
FPGA	Field Programmable Gate Array
IO	Input Output
IP	Intellectual Property
LUT	Look Up Table
PC	Personal Computer
RAM	Random Access Memory
SD	Secure Digital
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

2 IP Configuration and Instantiation

2.1 IP Configuration

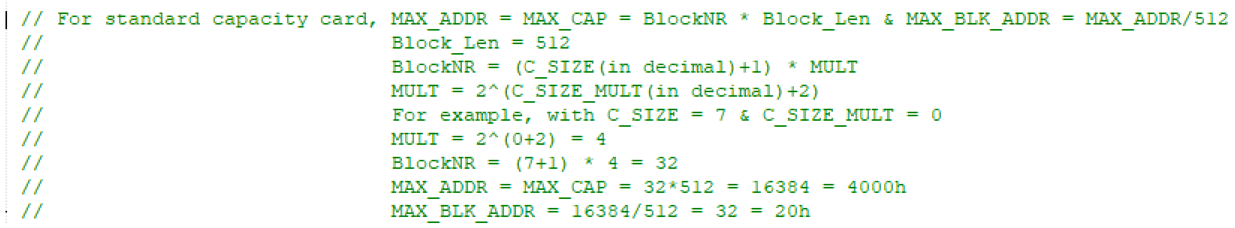
The SD Slave Controller IP is instantiated in the design as shown in the figure below.



```
work
  example_top (example_design.v)
    IO_DDR (IO_DDR.v)
    sd_mem_top (sd_mem_top.v)
      sd_cmd_path (sd_cmd_path.v)
      sd_data_path (sd_data_path.v)
      synchronizer (synchronizer.v)
      user_if (user_if.v)
    user_app (user_app.v)
      ram_2kx8 (ram_2kx8.v)
```

Figure 1 :Design sources where SD slave IP is instantiated

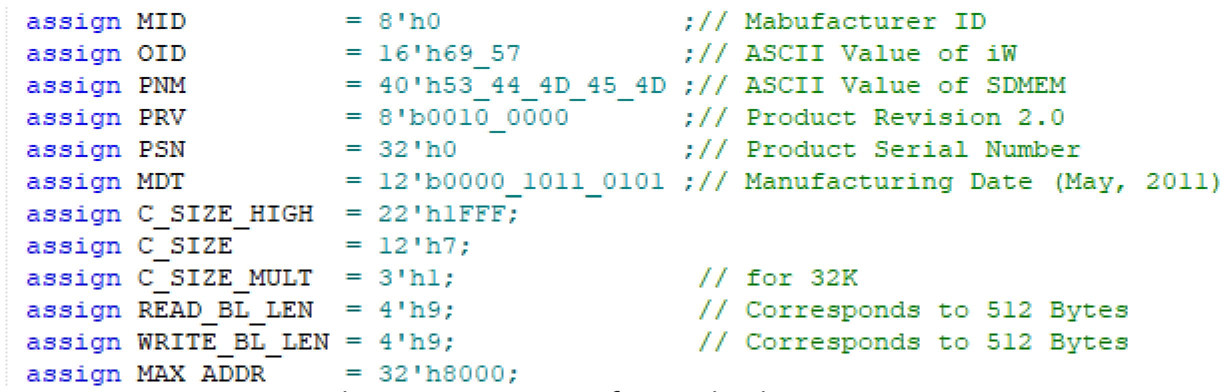
Module **example_top** is the top module of the project which integrates the top module of SD Slave IP and **user_app** module.The **sd_mem_top** module will integrate the **user_if** module with the **sd_cmd_path**, **sd_data_path** and the **synchronizer** modules. There are few SD parameters which can be modified by the user to change the size of the SD.



```
// For standard capacity card, MAX_ADDR = MAX_CAP = BlockNR * Block_Len & MAX_BLK_ADDR = MAX_ADDR/512
//                             Block_Len = 512
//                             BlockNR = (C_SIZE(in decimal)+1) * MULT
//                             MULT = 2^(C_SIZE_MULT(in decimal)+2)
//                             For example, with C_SIZE = 7 & C_SIZE_MULT = 0
//                             MULT = 2^(0+2) = 4
//                             BlockNR = (7+1) * 4 = 32
//                             MAX_ADDR = MAX_CAP = 32*512 = 16384 = 4000h
//                             MAX_BLK_ADDR = 16384/512 = 32 = 20h
```

Figure 2 :SD Size Calculation

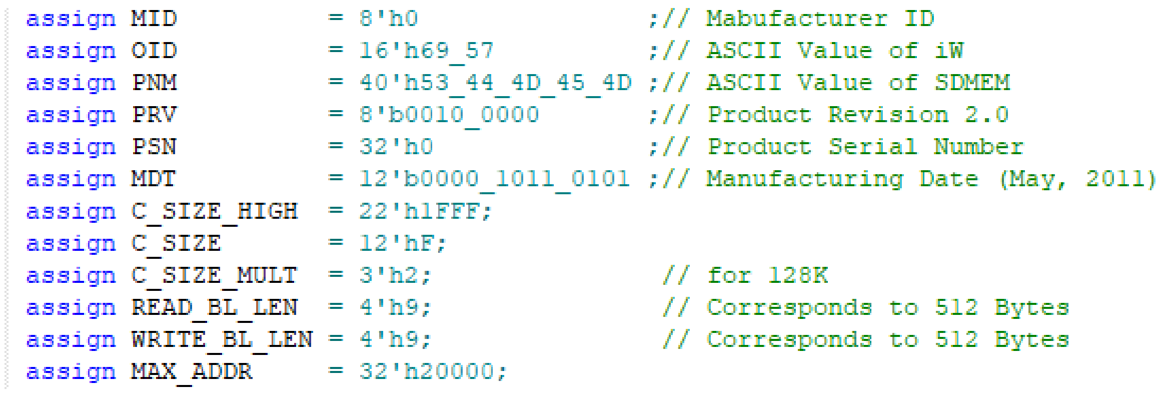
The above figure shows the calculation steps for the SD size of 16KiB.



```
assign MID          = 8'h0                ;// Mabufacturer ID
assign OID          = 16'h69_57           ;// ASCII Value of iW
assign PNM          = 40'h53_44_4D_45_4D ;// ASCII Value of SDMEM
assign PRV          = 8'b0010_0000        ;// Product Revision 2.0
assign PSN          = 32'h0               ;// Product Serial Number
assign MDT          = 12'b0000_1011_0101 ;// Manufacturing Date (May, 2011)
assign C_SIZE_HIGH  = 22'h1FFF;
assign C_SIZE       = 12'h7;
assign C_SIZE_MULT  = 3'h1;                 // for 32K
assign READ_BL_LEN  = 4'h9;                 // Corresponds to 512 Bytes
assign WRITE_BL_LEN = 4'h9;                 // Corresponds to 512 Bytes
assign MAX_ADDR     = 32'h8000;
```

Figure 3 :Parameters for 32KiB size SD

The figure above shows the configurable parameters in **example_top.v** file.Parameters like **C_SIZE**, **C_SIZE_MULT** and **MAX_ADDR** parameters which decides the size of the SD implemented are controlled directly from the example_top.v module as shown in the Figure 2. In Figure 3, parameters are given for the 128KiB SD based on the calculation steps given on the Figure 2.



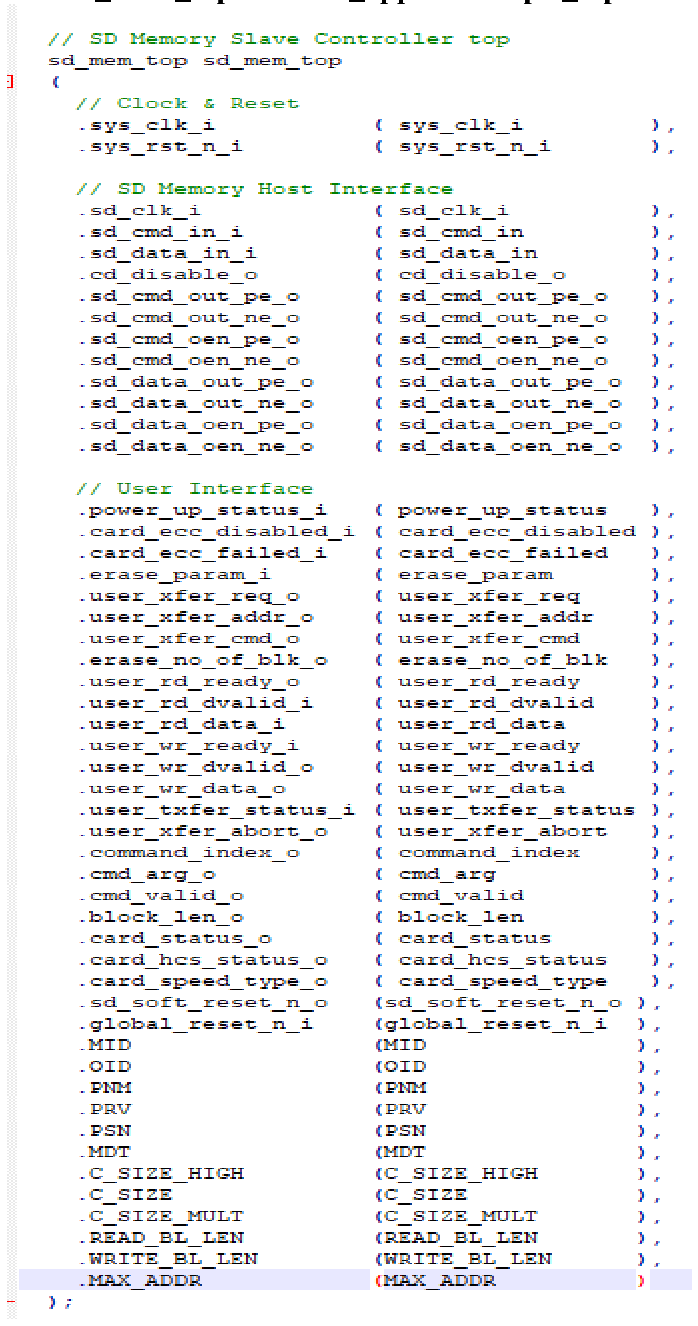
```
assign MID         = 8'h0              ;// Mabufacturer ID
assign OID         = 16'h69_57         ;// ASCII Value of iW
assign PNM         = 40'h53_44_4D_45_4D ;// ASCII Value of SDMEM
assign PRV         = 8'b0010_0000      ;// Product Revision 2.0
assign PSN         = 32'h0             ;// Product Serial Number
assign MDT         = 12'b0000_1011_0101 ;// Manufacturing Date (May, 2011)
assign C_SIZE_HIGH  = 22'h1FFF;
assign C_SIZE       = 12'hF;
assign C_SIZE_MULT  = 3'h2;             // for 128K
assign READ_BL_LEN  = 4'h9;             // Corresponds to 512 Bytes
assign WRITE_BL_LEN = 4'h9;             // Corresponds to 512 Bytes
assign MAX_ADDR     = 32'h20000;
```

Figure 4 :Parameters for 128KiB size SD

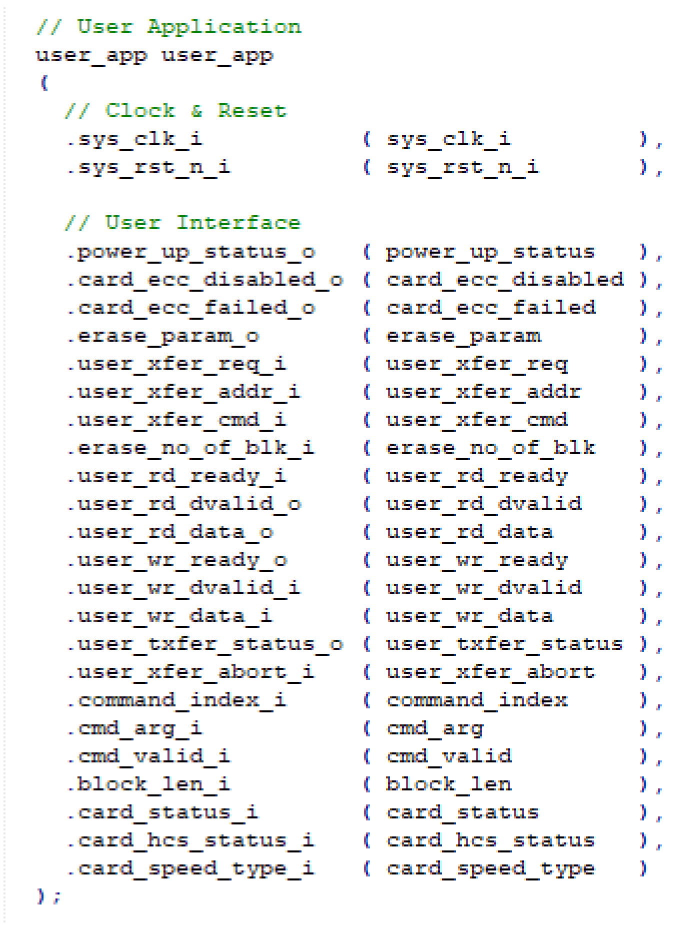
NOTE: Based on the values of **C_SIZE**, **C_SIZE_MULT** and **MAX_ADDR**, SD size is implemented. Maximum size of the SD that can be implemented depends on resources available in the device.

2.2 Steps to Instantiate SD Memory Slave IP

1) Instantiate the modules **sd_mem_top** and **user_app** in **example_top** as below,



```
// SD Memory Slave Controller top
sd_mem_top sd_mem_top
(
  // Clock & Reset
  .sys_clk_i             ( sys_clk_i          ),
  .sys_rst_n_i           ( sys_rst_n_i        ),

  // SD Memory Host Interface
  .sd_clk_i              ( sd_clk_i           ),
  .sd_cmd_in_i           ( sd_cmd_in          ),
  .sd_data_in_i          ( sd_data_in         ),
  .cd_disable_o          ( cd_disable_o       ),
  .sd_cmd_out_pe_o       ( sd_cmd_out_pe_o    ),
  .sd_cmd_out_ne_o       ( sd_cmd_out_ne_o    ),
  .sd_cmd_oen_pe_o       ( sd_cmd_oen_pe_o    ),
  .sd_cmd_oen_ne_o       ( sd_cmd_oen_ne_o    ),
  .sd_data_out_pe_o      ( sd_data_out_pe_o   ),
  .sd_data_out_ne_o      ( sd_data_out_ne_o   ),
  .sd_data_oen_pe_o      ( sd_data_oen_pe_o   ),
  .sd_data_oen_ne_o      ( sd_data_oen_ne_o   ),

  // User Interface
  .power_up_status_i     ( power_up_status    ),
  .card_ecc_disabled_i   ( card_ecc_disabled  ),
  .card_ecc_failed_i     ( card_ecc_failed    ),
  .erase_param_i         ( erase_param        ),
  .user_xfer_req_o       ( user_xfer_req      ),
  .user_xfer_addr_o      ( user_xfer_addr     ),
  .user_xfer_cmd_o       ( user_xfer_cmd      ),
  .erase_no_of_blk_o     ( erase_no_of_blk    ),
  .user_rd_ready_o       ( user_rd_ready      ),
  .user_rd_dvalid_i      ( user_rd_dvalid     ),
  .user_rd_data_i        ( user_rd_data       ),
  .user_wr_ready_i       ( user_wr_ready      ),
  .user_wr_dvalid_o      ( user_wr_dvalid     ),
  .user_wr_data_o        ( user_wr_data       ),
  .user_txfer_status_i   ( user_txfer_status  ),
  .user_xfer_abort_o     ( user_xfer_abort    ),
  .command_index_o       ( command_index      ),
  .cmd_arg_o             ( cmd_arg            ),
  .cmd_valid_o           ( cmd_valid          ),
  .block_len_o           ( block_len          ),
  .card_status_o         ( card_status        ),
  .card_hcs_status_o     ( card_hcs_status    ),
  .card_speed_type_o     ( card_speed_type    ),
  .sd_soft_reset_n_o     (sd_soft_reset_n_o ),
  .global_reset_n_i      (global_reset_n_i  ),
  .MID                   (MID               ),
  .OID                   (OID               ),
  .PNM                   (PNM               ),
  .PRV                   (PRV               ),
  .PSN                   (PSN               ),
  .MDT                   (MDT               ),
  .C_SIZE_HIGH           (C_SIZE_HIGH       ),
  .C_SIZE                (C_SIZE            ),
  .C_SIZE_MULT           (C_SIZE_MULT       ),
  .READ_BL_LEN           (READ_BL_LEN       ),
  .WRITE_BL_LEN          (WRITE_BL_LEN      ),
  .MAX_ADDR              (MAX_ADDR          )
);
```

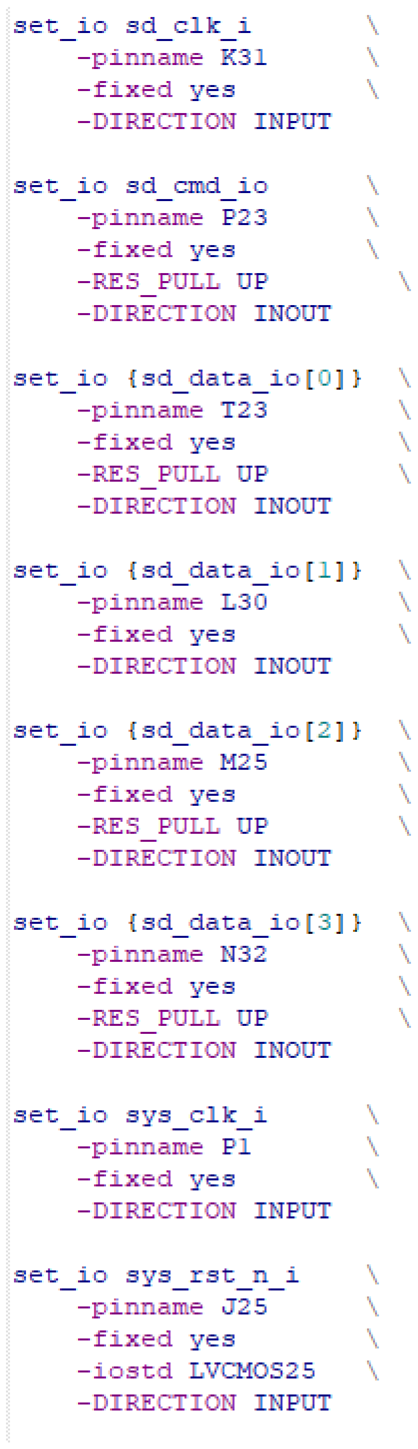


```
// User Application
user_app user_app
(
  // Clock & Reset
  .sys_clk_i            ( sys_clk_i         ),
  .sys_rst_n_i          ( sys_rst_n_i       ),

  // User Interface
  .power_up_status_o    ( power_up_status   ),
  .card_ecc_disabled_o  ( card_ecc_disabled ),
  .card_ecc_failed_o    ( card_ecc_failed   ),
  .erase_param_o        ( erase_param       ),
  .user_xfer_req_i      ( user_xfer_req     ),
  .user_xfer_addr_i     ( user_xfer_addr    ),
  .user_xfer_cmd_i      ( user_xfer_cmd     ),
  .erase_no_of_blk_i    ( erase_no_of_blk   ),
  .user_rd_ready_i      ( user_rd_ready     ),
  .user_rd_dvalid_o     ( user_rd_dvalid    ),
  .user_rd_data_o       ( user_rd_data      ),
  .user_wr_ready_o      ( user_wr_ready     ),
  .user_wr_dvalid_i     ( user_wr_dvalid    ),
  .user_wr_data_i       ( user_wr_data      ),
  .user_txfer_status_o  ( user_txfer_status ),
  .user_xfer_abort_i    ( user_xfer_abort   ),
  .command_index_i      ( command_index     ),
  .cmd_arg_i            ( cmd_arg           ),
  .cmd_valid_i          ( cmd_valid         ),
  .block_len_i          ( block_len         ),
  .card_status_i        ( card_status       ),
  .card_hcs_status_i    ( card_hcs_status   ),
  .card_speed_type_i    ( card_speed_type   )
);
```

Figure 5: Instantiation of sd_mem_top and user_app top modules

example_top.v is the top module of SD Slave IP which is used to integrate the **sd_mem_top.v** module with **user_app.v** module.

2) The Constraint file (**.pdc**) provided with the design can be changed by user depending on board and requirements.
3) **sd_mem_define.v** file is the header file of the SD Slave IP.



```
set_io sd_clk_i        \
    -pinname K31       \
    -fixed yes         \
    -DIRECTION INPUT

set_io sd_cmd_io       \
    -pinname P23       \
    -fixed yes         \
    -RES_PULL UP         \
    -DIRECTION INOUT

set_io {sd_data_io[0]}  \
    -pinname T23        \
    -fixed yes          \
    -RES_PULL UP        \
    -DIRECTION INOUT

set_io {sd_data_io[1]}  \
    -pinname L30        \
    -fixed yes          \
    -DIRECTION INOUT

set_io {sd_data_io[2]}  \
    -pinname M25        \
    -fixed yes          \
    -RES_PULL UP        \
    -DIRECTION INOUT

set_io {sd_data_io[3]}  \
    -pinname N32        \
    -fixed yes          \
    -RES_PULL UP        \
    -DIRECTION INOUT

set_io sys_clk_i       \
    -pinname P1        \
    -fixed yes         \
    -DIRECTION INPUT

set_io sys_rst_n_i     \
    -pinname J25       \
    -fixed yes         \
    -iostd LVCMOS25    \
    -DIRECTION INPUT
```

Figure 6: SD Slave Interface Constraints in pinout .pdc file

NOTE: In the **pinout.pdc** file, the pin constraints are applied with respect to the test board. Please change the pin constraints for SD interface as required for custom design.

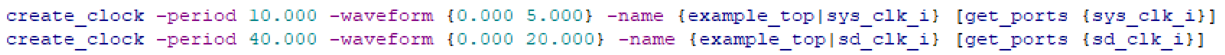
3 Implementation Details

3.1 Clock Domain

SD Slave Controller IP works at clock speed 25Mhz for default speed configuration. System clock input whose frequency is 100Mhz.

3.2 Constraints

The following figure shows the clock constraints used in this design. The constraints given here the system clock whose frequency is 100Mhz and SD clock whose frequency is 25Mhz for default speed. It is defined in the **example_top_sdc.sdc** file



```
create_clock -period 10.000 -waveform {0.000 5.000} -name {example_top|sys_clk_i} [get_ports {sys_clk_i}]
create_clock -period 40.000 -waveform {0.000 20.000} -name {example_top|sd_clk_i} [get_ports {sd_clk_i}]
```

Figure 7: Clock constraints

NOTE: Please change the clock constraints as required for custom design. As also choose the required IO standards

4 Test Setup

4.1 Scope

This section describes the Hardware connection procedure to power-on, inserting SD Memory slave demo board.

4.2 Overview

SD Host is used to evaluate SD Slave controller IP. It has SD slot in it by which we can test iW-SD memory slave IP by connecting the SD extension cable.

4.3 Boards and Accessories

- SmartFusion2 Advanced Development Kit
- SD Host
- FMC Card with SD Slot
- SD extension cable
- 1 USB A to mini–B Cable for Programming
- 1 USB cable for UART

4.4 Software Requirements

- A Laptop/PC with Tera Term installed
- Microsemi Smart Debug or Libero 11.9 SP6

4.5 Connections

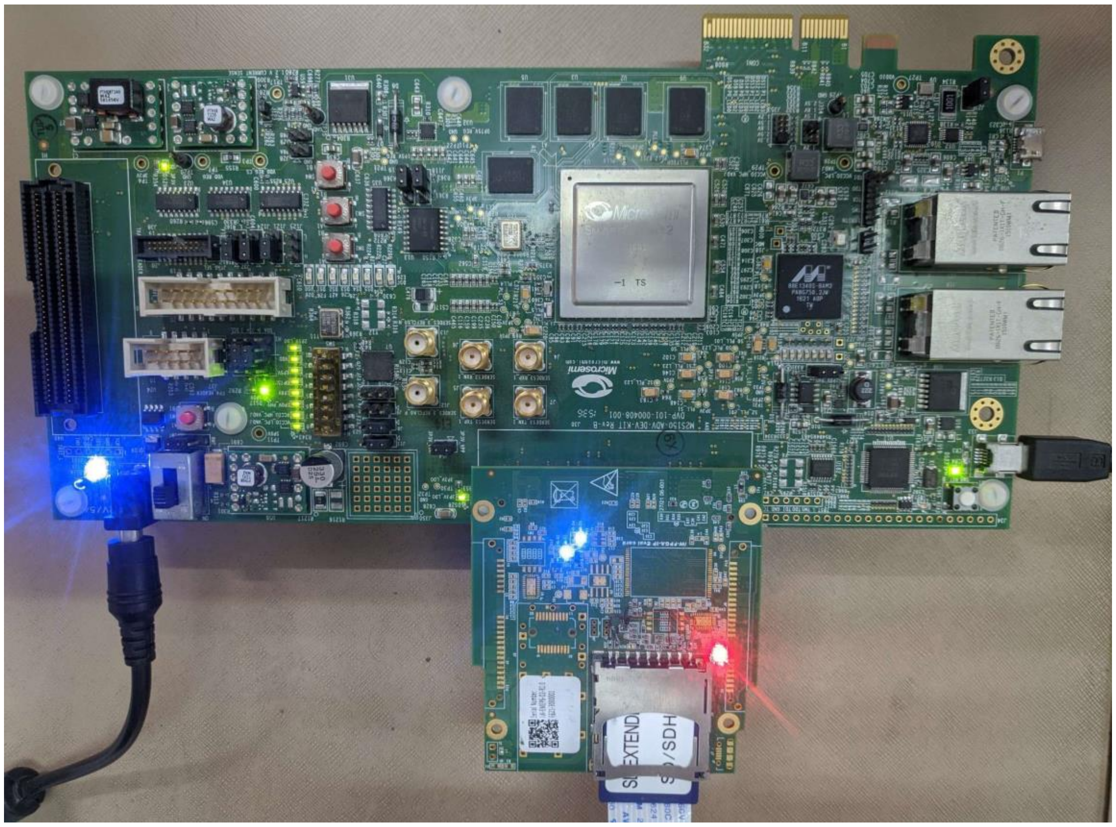


Figure 8: Test Setup with FMC and SD Extension Cable

Do the connections as shown in above figure

- Connect SD extension cable to SD Card Slot in FMC card.
- 1 USB A to mini–B Cable for Programming

4.6 Programming FPGA using the Microsemi Smart Debug

- Open the Smart Debug Tool
- Create a New Project from the Project option.
- Give Name and the location where the project to be created, select Construct Automatically option and Click OK.
- Click on Programming Connectivity and Interface Icon (3rd Icon), select "YES" in the pop-up warning.
- Right Click a device in the Programming Connectivity and Interface dialog box.
- Select Load Programming File and select the **.stp** file provided.
- After Programming is completed press the **SW1** (below the power on switch) push button to reset the board once.

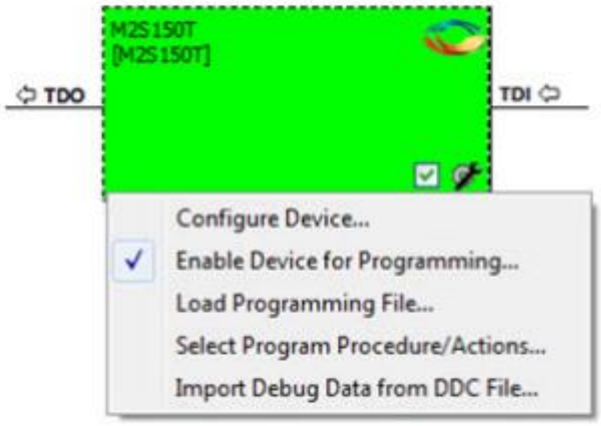
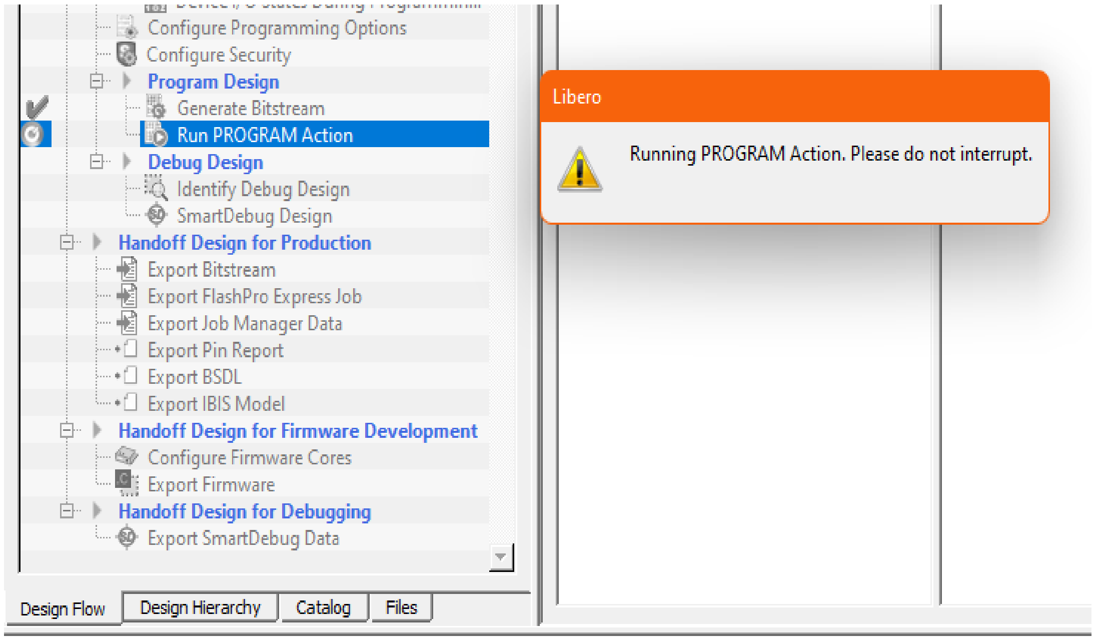

M2S150T
[M2S150T]
TDO
TDI
Configure Device...
Enable Device for Programming...
Load Programming File...
Select Program Procedure/Actions...
Import Debug Data from DDC File...

Figure 9: Programming the Board from Smart Debug

4.7 Programming FPGA using the Libero Tool

- Open the FPGA project in the Libero Tool
- Compile the project and Generate the Bitstream, if there is any change in the design.
- Click on the Run Program Action option under the Generate Bitstream Option.
- Wait until the Bitstream is programmed completely.
- After Programming is completed press the **SW1** (below the power on switch) push button to reset the board once.



Running PROGRAM Action. Please do not interrupt.

Figure 10: Programming the Board from Smart Debug

5 Test Procedure

5.1 Tera Term Setup

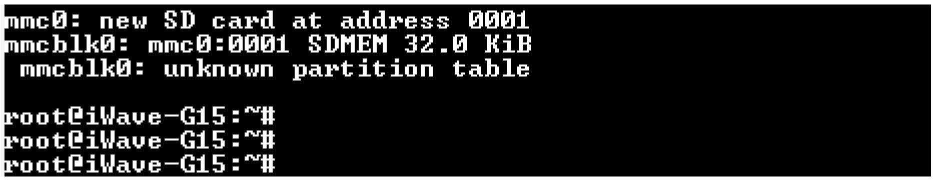
- Power the SD Host and connect it to PC/Laptop via com port
- Open tera term in Laptop/PC and select the com port
- Select baud rate as 115200

5.2 Testing in SD Host

After SD Host is completely boot, then follow the below procedure to test the SD slave IP.

5.3 Card Insertion Test on SD Host

Insert the SD extension cable from test board to SD Host. When the extension card is inserted check the tera term. The terminal print is as shown in below figure.



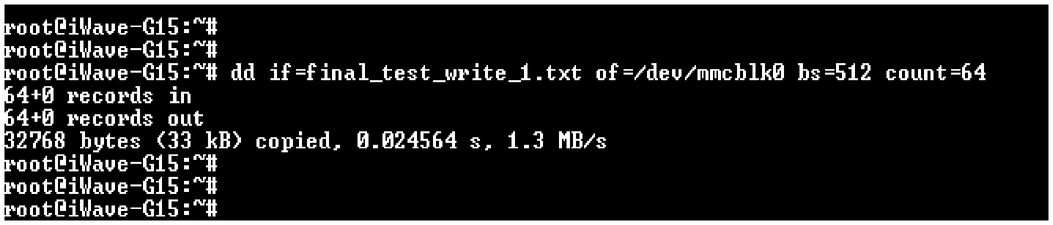
```
mmc0: new SD card at address 0001
mmcblk0: mmc0:0001 SDMEM 32.0 KiB
 mmcblk0: unknown partition table

root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~#
```

Figure 11: Card Detection Prints for default speed

This Indicates that 32KiB SD Slave is detected successfully at the default speed.

5.3.1 Testing Read and Write using the ‘dd’ commands

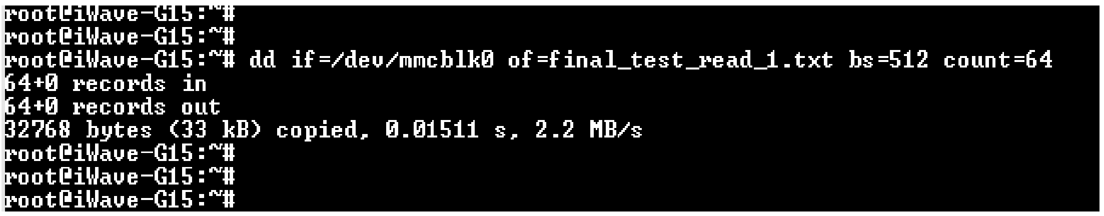
Create a file of 32 KiB size **final_test_write_1.txt**.
Once the card is detected on the SD Host, use the commands shown below to write the “**final_test_write_1.txt**” file into the SD card.



```
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~# dd if=final_test_write_1.txt of=/dev/mmcblk0 bs=512 count=64
64+0 records in
64+0 records out
32768 bytes (33 kB) copied, 0.024564 s, 1.3 MB/s
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~#
```

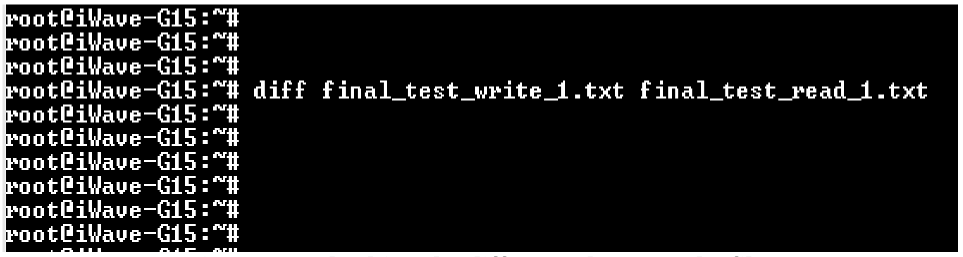
Figure 12: Writing to the Card using the “dd” command

Once the writing is completed, use the “**dd**” command as shown below to readback the file written to the SD card.



```
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~# dd if=/dev/mmcblk0 of=final_test_read_1.txt bs=512 count=64
64+0 records in
64+0 records out
32768 bytes (33 kB) copied, 0.01511 s, 2.2 MB/s
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~#
```

Figure 13: Reading from the Card using the "dd" command



```
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~# diff final_test_write_1.txt final_test_read_1.txt
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~#
```

Figure 14: Checking the difference between the files

The **"diff"** command is used to check the difference between the file written to the SD Slave and the file readback from the SD Slave.

On executing the "**diff**" command, if there is no difference in those files, nothing is printed on the terminal.
If there is any difference, then the difference is returned on the terminal.

6 Simulation

6.1 Scope

This section describes how SD Memory Slave is simulated so that it meets the functionality.

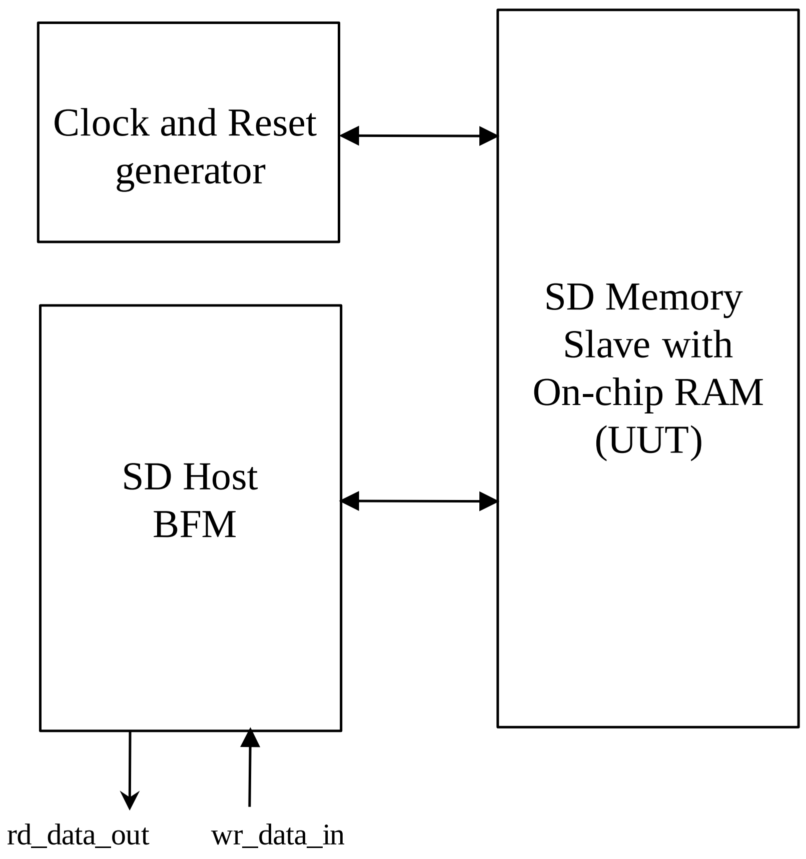

Clock and Reset generator
SD Memory Slave with On-chip RAM (UUT)
SD Host BFM
rd_data_out
wr_data_in

Figure 15: Test Environment Block Diagram

6.2 Description of Test Environment:

The test environment consists of mainly below units:

- Clock and Reset generator
- SD Host BFM
- Text files

- **SD memory Slave IP (UUT):** SD memory slave IP acts as the SD memory device which is the unit under test.
- **Clock and Reset generator:** This module will generate the system clock and system reset signal required for the SD memory IP.

- **SD Host BFM:**

SD host BFM has the different tasks which will generate the different commands to test the IP. This module also receives the response from the IP and checks whether the response is received properly or not.
During the write operation this BFM reads the data from the wr_data_in.txt and passes the data in the data line as the write data to the IP.
During read it receives the data from IP and compares with the input data and displays test result depending on the result of comparison.

- **Test case Description:**

The defined test case will be used to test the single and multi-block write and read. The test case will first test the single block operation by writing the 512 bytes of the data followed by the read of same. After the successful write and read the host BFM will compare the read data with written data. If there is no data mismatch which means the test is successful.

Then BFM will initiate the multi-block write with 4 blocks of data followed by the multi-block read. After completion of all block write and read, the BFM compares the read data with written data and depending on the comparison, the test result will be displayed. User can check the waveform and confirm that the test results.

6.3 Simulation Steps

The following steps are followed to simulate the design in ModelSim.

1. Open ModelSim and change directory to the simulation folder with the following command in the ModelSim prompt.
ModelSim> cd {*\iW-PRHHJ-FF-01-R1.0-REL1.2\iW-PRHHJ-SI-01-R1.0-REL1.0\iW-PRHHJ-SI-01-R1.0_REL1.0_Functional_4bit.zip}

2. Simulation for a given test case can be carried out with the command ModelSim> do run.do

3. The run.do script will run the simulation and the results can be checked in the ModelSim Terminal.

4. Make sure that Single Block and Multi Block Tests are Passed from the ModelSim Terminal.

Note 1: '*' indicates the release package directory in PC.
Note 2: Current Test environment is simulated in ModelSim SE 10.4d.

7 Resource Utilization

The table below shows the resource utilization summary for **SmartFusion2 M2S150TS-1152FC-1** device with SD Memory Slave Controller IP.

Table 2: Resource Utilization

Resource	Utilization	Available	Utilization %
LUT	4861	146124	3.33
DFF	4204	146124	2.88
RAM1K18	37	236	15.68
RAM64x18	8	240	3.33
IO	8	574	1.39