SD/SDIO Host Controller 3.0 IP Integration Manual

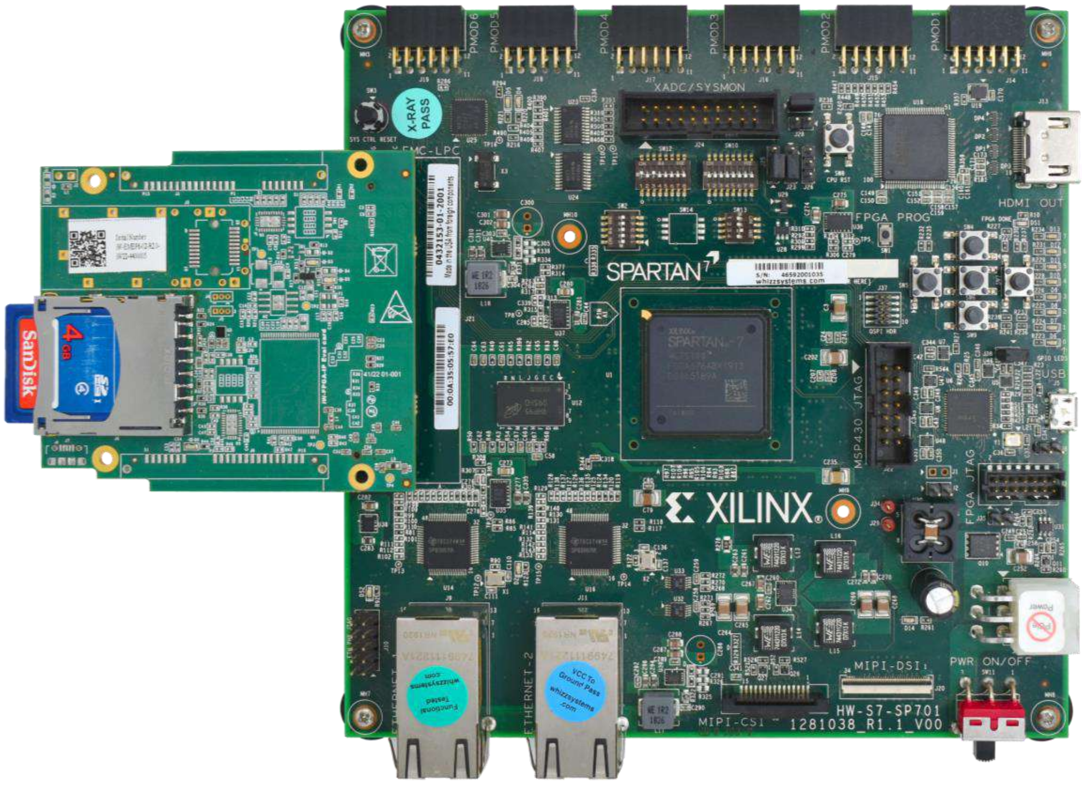


Table of Content

List Of Figures

List Of Tables

1 Introduction

1.1 Purpose

The purpose of this document is to describe SD / SDIO Host Controller 3.0. IP Integration details.

1.2 Reference Document

- IP datasheet

1.3 Overview

SD/SDIO Host Controller interfaces the Microblaze through AXI4 bus. And MiG Controller through AXI memory mapped interface, enabling the data transfers between each other. Microblaze IP will send the response to the SD host depending on the command issued and also communicates with the MIG controller for Data read & write.

1.4 Acronyms and Abbreviations

Table 1: Acronyms & Abbreviations

Term	Meaning
FPGA	Field Programmable Gate Array
GPIO	General purpose input output
FMC	FPGA Mezzanine Card
LUT	Look Up Table
IO	Input and Output
FF	Flip Flop

2 IP Configuration and Instantiation

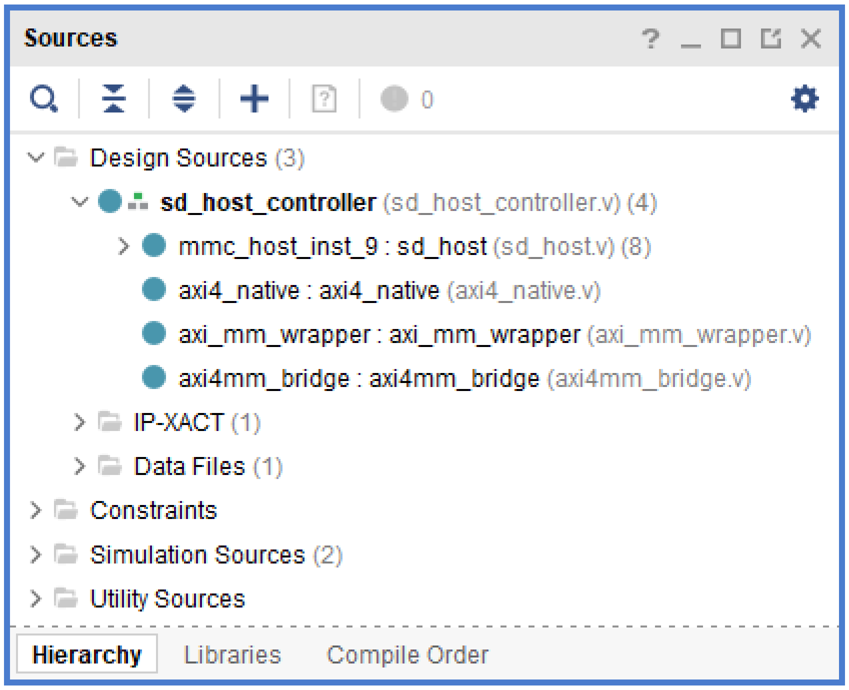
2.1 Example design

The SD/SDIO host controller IP example design mainly consists of,

1. **Microblaze soft core:** User can provide the inputs to run the different tests using the bare metal code running in the Microblaze. This will pass the different control signals to SD host controller based on the user selection.
2. **SD/SDIO Host Controller IP:** SD host controller IP takes the command given from the microblaze soft core to the register set and with the ADMA the read and write operation will happen from and to SD Card

2.2 IP Configuration

The SD Host Controller IP is configured before synthesis. The register set, adma configuration , tuning block, command and data files sd_host file is instantiated in the design as shown in the figure for generating the SD Host controller IP. Other files the axi4_lite interface and the axi memory mapped interface wrapper files are added to make a user configured IP to be easy add in the block design.


Sources

Design Sources (3)
sd_host_controller (sd_host_controller.v) (4)
mmc_host_inst_9 : sd_host (sd_host.v) (8)
axi4_native : axi4_native (axi4_native.v)
axi_mm_wrapper : axi_mm_wrapper (axi_mm_wrapper.v)
axi4mm_bridge : axi4mm_bridge (axi4mm_bridge.v)
IP-XACT (1)
Data Files (1)
Constraints
Simulation Sources (2)
Utility Sources

Figure 2-1:Design sources where SD host is instantiated

Module sd_host_controller is the top module of the project. The sd_host_controller IP needs to be added in the block design of the Vivado project 2020.1.

2.3 Steps to add SD HOST IP to the Block design

- Install the required Vivado Design Suite 2020.1 for the host PC adding the license path. Downloads (xilinx.com)
- Copy the project from "….\EMFDY_Release1.3_SD_3.0_Host\iW-EMFDY-PF-01-R1.0-REL1.3\iW-EMFDY-FF-01-R1.0-REL1.3\iW-EMFDY-ED-01-R1.0-REL1.3\ sd_host_proj_190MHz" to your project directory in the host PC.
- Open the Xilinx Vivado tool 2020.1 and click on "Open Project" and select the required project from the folder by clicking ok.
- Now the project selected will be opened in Vivado Design Suite 2020.1.
- Go to settings of Vivado 2020.1 and add the IP to the Project Settings → IP → Repository→then add the IP path .

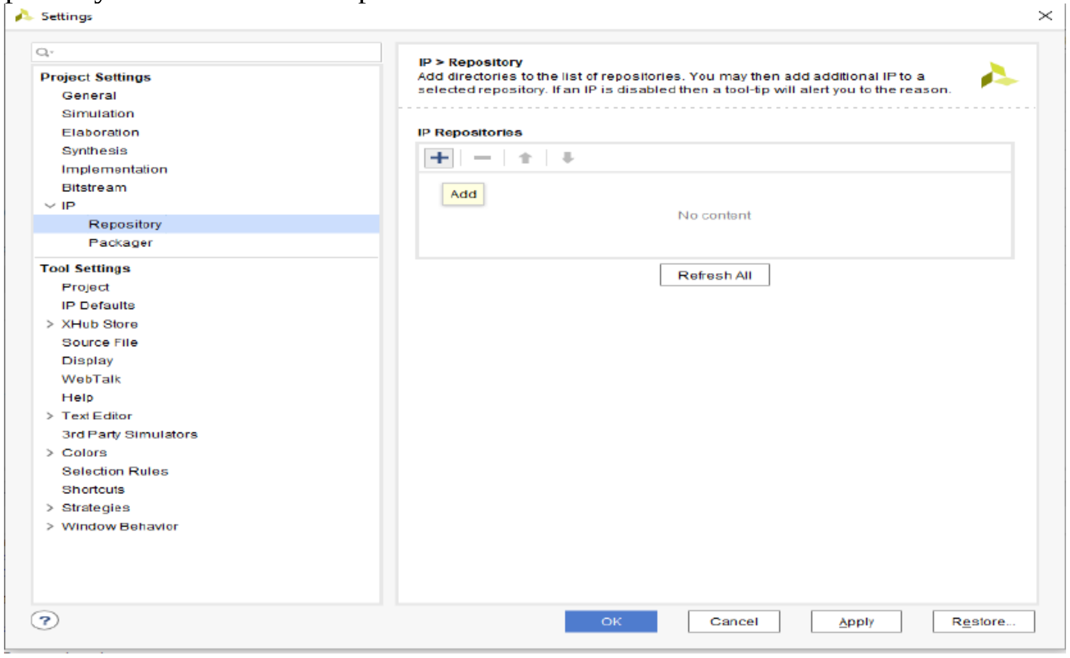
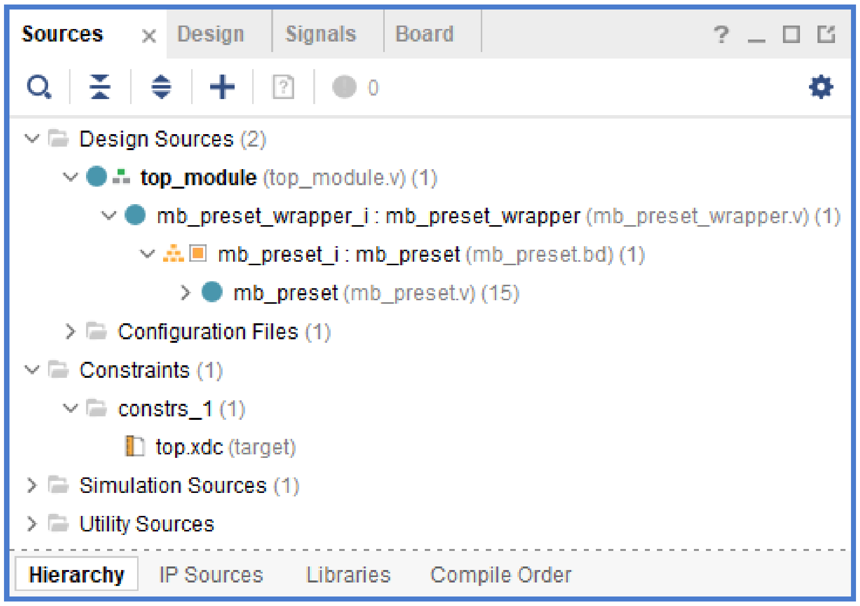

IP > Repository
Add directories to the list of repositories. You may then add additional IP to a selected repository. If an IP is disabled then a tool-tip will alert you to the reason.

Figure 2-2:Adding SD_Host_controller IP to the IP repository

IP location is ..\EMFDY_Release1.3_SD_3.0_Host\iW-EMFDY-PF-01-R1.0-REL1.3\iW-EMFDY-FF-01-R1.0-REL1.3\iW-EMFDY-ED-01-R1.0-REL1.3\sd_host_proj_190MHz \sd_host_project.srcs \sources_1 \imports\IP.Add IP by clicking on the + sign as shown in Figure above. After the IP is added click on Apply and close the dialogue box

Once the IP is added in the project repository open the block design by clicking on mb _preset under top_module → mb_preset_wrapper present in the source tab.



Sources × Design Signals Board

0

Design Sources (2)
- top_module (top_module.v) (1)
  - mb_preset_wrapper_i : mb_preset_wrapper (mb_preset_wrapper.v) (1)
    - mb_preset_i : mb_preset (mb_preset.bd) (1)
      - mb_preset (mb_preset.v) (15)
- Configuration Files (1)

Constraints (1)
- constrs_1 (1)
  - top.xdc (target)

Simulation Sources (1)

Utility Sources

Hierarchy IP Sources Libraries Compile Order

Figure 2-3:Click on Open Block design

1. After adding the other IPs like Microblaze ,UART ,MIG Controller and Clocking Wizard, add the SD host controller IP to the block design by clicking on the + sign as shown in the Figure below

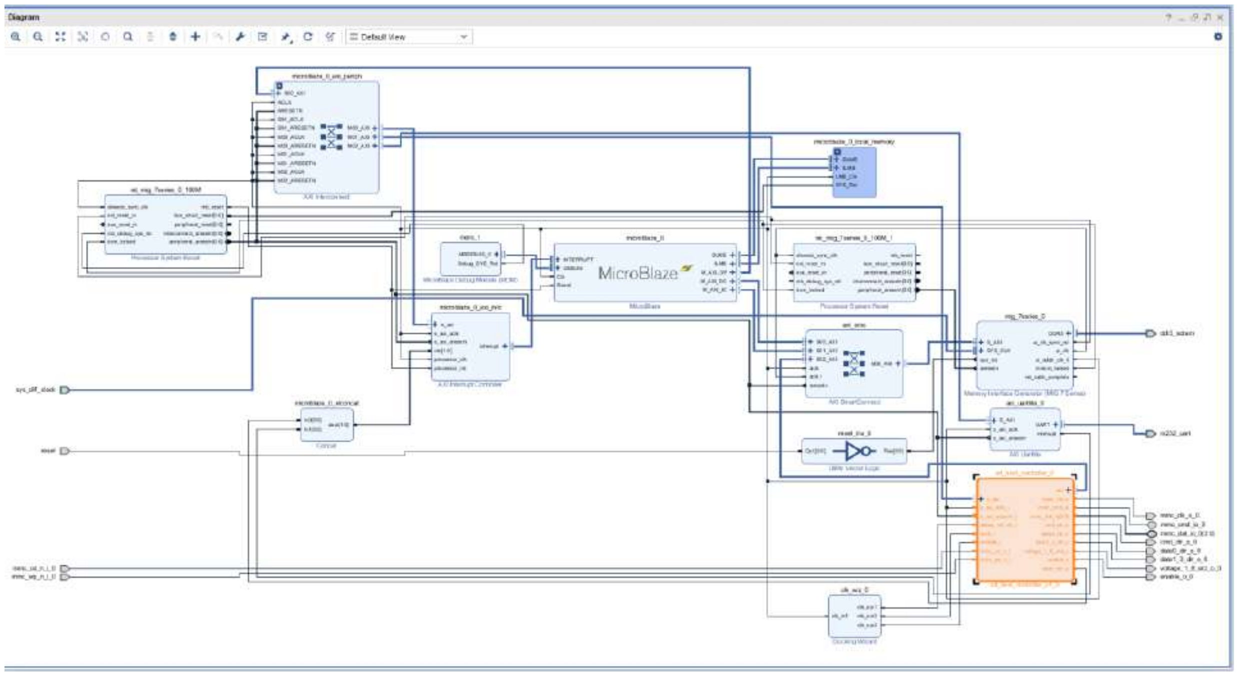
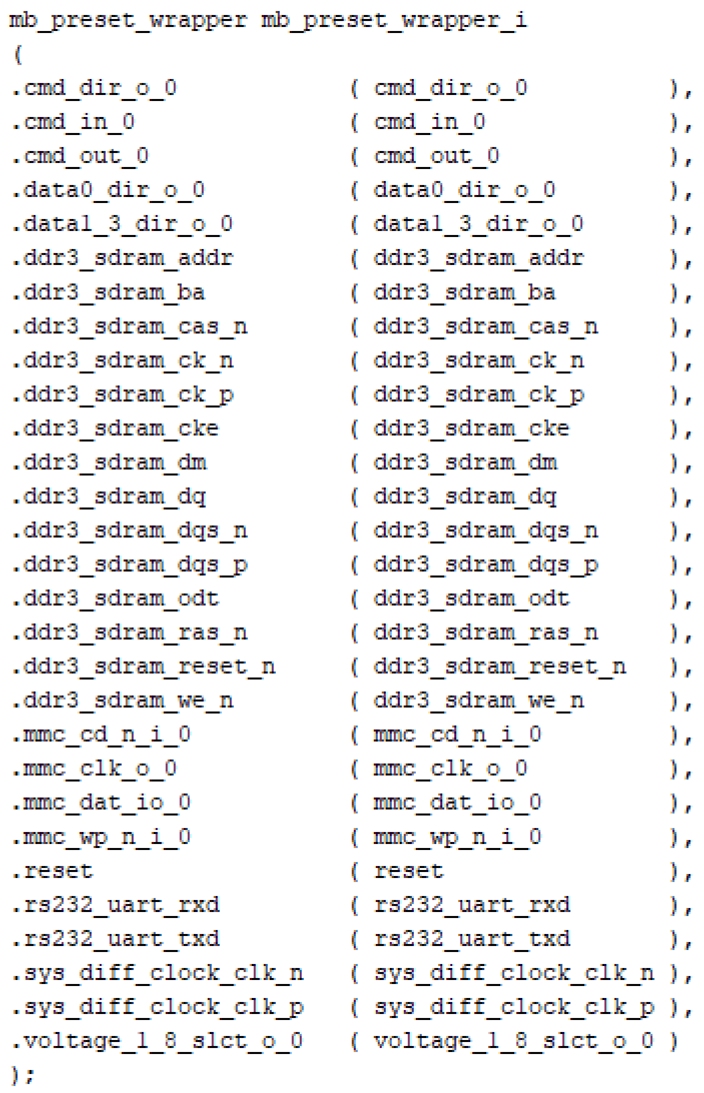


Figure 2-4:SD Host Controller added to the block design

2. Synthesize the block design first by validating the design(F6) and generate the output products by right clicking on mb_preset_wrapper under top_module → mb_preset_wrapper present in the source tab and create HDL Wrapper
3. Instantiate mb_preset_wrapper in the top module with the ports and signals of block design, in the top_module.v file as shown in the Figure below



```
mb_preset_wrapper mb_preset_wrapper_i
(
.cmd_dir_o_0            ( cmd_dir_o_0          ),
.cmd_in_0               ( cmd_in_0             ),
.cmd_out_0              ( cmd_out_0            ),
.data0_dir_o_0          ( data0_dir_o_0        ),
.data1_3_dir_o_0        ( data1_3_dir_o_0      ),
.ddr3_sdram_addr        ( ddr3_sdram_addr      ),
.ddr3_sdram_ba          ( ddr3_sdram_ba        ),
.ddr3_sdram_cas_n       ( ddr3_sdram_cas_n     ),
.ddr3_sdram_ck_n        ( ddr3_sdram_ck_n      ),
.ddr3_sdram_ck_p        ( ddr3_sdram_ck_p      ),
.ddr3_sdram_cke         ( ddr3_sdram_cke       ),
.ddr3_sdram_dm          ( ddr3_sdram_dm        ),
.ddr3_sdram_dq          ( ddr3_sdram_dq        ),
.ddr3_sdram_dqs_n       ( ddr3_sdram_dqs_n     ),
.ddr3_sdram_dqs_p       ( ddr3_sdram_dqs_p     ),
.ddr3_sdram_odt         ( ddr3_sdram_odt       ),
.ddr3_sdram_ras_n       ( ddr3_sdram_ras_n     ),
.ddr3_sdram_reset_n     ( ddr3_sdram_reset_n   ),
.ddr3_sdram_we_n        ( ddr3_sdram_we_n      ),
.mmc_cd_n_i_0           ( mmc_cd_n_i_0         ),
.mmc_clk_o_0            ( mmc_clk_o_0          ),
.mmc_dat_io_0           ( mmc_dat_io_0         ),
.mmc_wp_n_i_0           ( mmc_wp_n_i_0         ),
.reset                  ( reset                ),
.rs232_uart_rxd         ( rs232_uart_rxd       ),
.rs232_uart_txd         ( rs232_uart_txd       ),
.sys_diff_clock_clk_n   ( sys_diff_clock_clk_n ),
.sys_diff_clock_clk_p   ( sys_diff_clock_clk_p ),
.voltage_1_8_slct_o_0   ( voltage_1_8_slct_o_0 )
);
```

Figure 2-5:Instantiation of block design IO signals in the top module

4. The Constraint file (.xdc) provided in the design is for SP701 dev. kit and should be changed for custom boards
5. Give the required clock, Pin/IO constraints for SD host controller in the .xdc file and compile the custom design with SD host IP.

3 Implementation Details

3.1 Clock Domain

SD Host Controller 3.0 IP works on the SDR104 mode and the base clock mclk_i drives the output clock to the SD card. User may need to adjust this clock between 104 MHz to 208 MHz based on the hardware setup. In other words, if user gets CRC error with say 200 MHz, then user can reduce the clock to say 190 MHz or below than to make sure write and read works fine with that base clock.

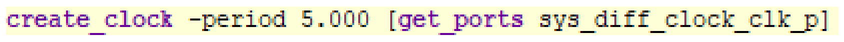
mclk90_i is a phase shifted version of the mclk_i and this phase shift may need to be tuned if we don't get the response for the CMD19. The phase shift will be initially kept 0 and later the phase shift can be added to this clock until we get the response for CMD19 from card.
Idelay_ref_clk_i is the fixed 200 MHz clock which will be provided for the IDELAY primitive to be used for the tuning implementation block.
s_axi_clk_i or hclk_i which is 100 MHz which will be used for the communication between the micro blaze and IP using the AXI4-lite interface for register access and AXI4-MM for DMA access.

So to generate the all these clocks we give a 200MHz on board clock to MIG controller and 100 MHz is generated from the MIG and that is given to the clocking wizard to generate other required clocks.

3.2 Constraints

3.2.1 Clock constraints:

The following figure shows the clock constraints used in this design. The constraints given here the system clock whose frequency is 200Mhz.
This constraint is added in the .xdc file. All other generated clocks will be automatically handled by the Vivado tool since those are derived from the MMCM and will have known relationship. User need to make sure there is no unconstrainted clock in the design by checking the clock network report from Vivado tool.

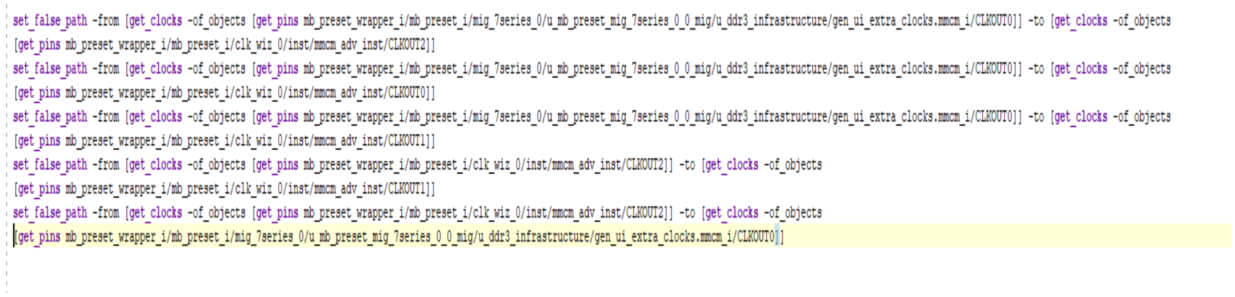


```
create_clock -period 5.000 [get_ports sys_diff_clock_clk_p]
```

Figure 3-1:Clock constraints

3.2.2 False path constraints:

The below figure shows the false path in between all the clocks which is configured for the SD host IP. This details will be available in the .xdc file.

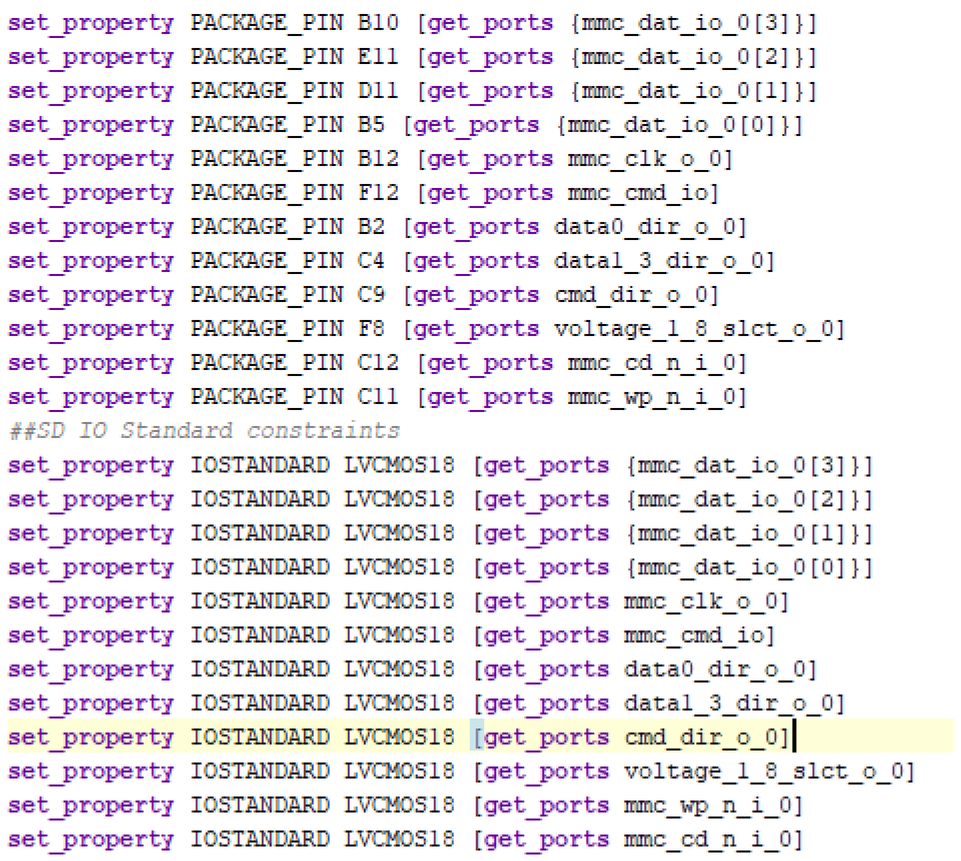


```
set_false_path -from [get_clocks -of_objects [get_pins mb_preset_wrapper_i/mb_preset_i/mig_7series_0/u_mb_preset_mig_7series_0_0_mig/u_ddr3_infrastructure/gen_ui_extra_clocks.mmcm_i/CLKOUT0]] -to [get_clocks -of_objects
[get_pins mb_preset_wrapper_i/mb_preset_i/clk_wiz_0/inst/mmcm_adv_inst/CLKOUT2]]
set_false_path -from [get_clocks -of_objects [get_pins mb_preset_wrapper_i/mb_preset_i/mig_7series_0/u_mb_preset_mig_7series_0_0_mig/u_ddr3_infrastructure/gen_ui_extra_clocks.mmcm_i/CLKOUT0]] -to [get_clocks -of_objects
[get_pins mb_preset_wrapper_i/mb_preset_i/clk_wiz_0/inst/mmcm_adv_inst/CLKOUT0]]
set_false_path -from [get_clocks -of_objects [get_pins mb_preset_wrapper_i/mb_preset_i/mig_7series_0/u_mb_preset_mig_7series_0_0_mig/u_ddr3_infrastructure/gen_ui_extra_clocks.mmcm_i/CLKOUT0]] -to [get_clocks -of_objects
[get_pins mb_preset_wrapper_i/mb_preset_i/clk_wiz_0/inst/mmcm_adv_inst/CLKOUT1]]
set_false_path -from [get_clocks -of_objects [get_pins mb_preset_wrapper_i/mb_preset_i/clk_wiz_0/inst/mmcm_adv_inst/CLKOUT2]] -to [get_clocks -of_objects
[get_pins mb_preset_wrapper_i/mb_preset_i/clk_wiz_0/inst/mmcm_adv_inst/CLKOUT1]]
set_false_path -from [get_clocks -of_objects [get_pins mb_preset_wrapper_i/mb_preset_i/clk_wiz_0/inst/mmcm_adv_inst/CLKOUT2]] -to [get_clocks -of_objects
[get_pins mb_preset_wrapper_i/mb_preset_i/mig_7series_0/u_mb_preset_mig_7series_0_0_mig/u_ddr3_infrastructure/gen_ui_extra_clocks.mmcm_i/CLKOUT0]]
```

Figure 3-2:False path constraints

3.2.3 Pin Constraints:

Below figure shows the example pin and IO standard constraints.



```
set_property PACKAGE_PIN B10 [get_ports {mmc_dat_io_0[3]}]
set_property PACKAGE_PIN E11 [get_ports {mmc_dat_io_0[2]}]
set_property PACKAGE_PIN D11 [get_ports {mmc_dat_io_0[1]}]
set_property PACKAGE_PIN B5 [get_ports {mmc_dat_io_0[0]}]
set_property PACKAGE_PIN B12 [get_ports mmc_clk_o_0]
set_property PACKAGE_PIN F12 [get_ports mmc_cmd_io]
set_property PACKAGE_PIN B2 [get_ports data0_dir_o_0]
set_property PACKAGE_PIN C4 [get_ports data1_3_dir_o_0]
set_property PACKAGE_PIN C9 [get_ports cmd_dir_o_0]
set_property PACKAGE_PIN F8 [get_ports voltage_1_8_slct_o_0]
set_property PACKAGE_PIN C12 [get_ports mmc_cd_n_i_0]
set_property PACKAGE_PIN C11 [get_ports mmc_wp_n_i_0]
##SD IO Standard constraints
set_property IOSTANDARD LVCMOS18 [get_ports {mmc_dat_io_0[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports {mmc_dat_io_0[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {mmc_dat_io_0[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {mmc_dat_io_0[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports mmc_clk_o_0]
set_property IOSTANDARD LVCMOS18 [get_ports mmc_cmd_io]
set_property IOSTANDARD LVCMOS18 [get_ports data0_dir_o_0]
set_property IOSTANDARD LVCMOS18 [get_ports data1_3_dir_o_0]
set_property IOSTANDARD LVCMOS18 [get_ports cmd_dir_o_0]
set_property IOSTANDARD LVCMOS18 [get_ports voltage_1_8_slct_o_0]
set_property IOSTANDARD LVCMOS18 [get_ports mmc_wp_n_i_0]
set_property IOSTANDARD LVCMOS18 [get_ports mmc_cd_n_i_0]
```

Figure 3-3:Pin Constraints in example design .xdc file

NOTE: These constraints are in accordance to example design created for the SP701 Evaluation kit and there is no need of adding the MIG/ DDR IO pins, UART IO pins .
Define the constraints for clock, reset and UART pins accordingly for any other custom board.

4 Test setup

4.1 Test requirements

- Xilinx SP701 dev. kit
- iWave FMC daughter card
- USB cables for JTAG and UART
- SD card for testing
- Power Supply 12 V,5A
  NOTE: iWave FMC daughter card requires V_{adj} =1.8V for its operation and the V_{adj} value depends on the custom board

4.2 Connections

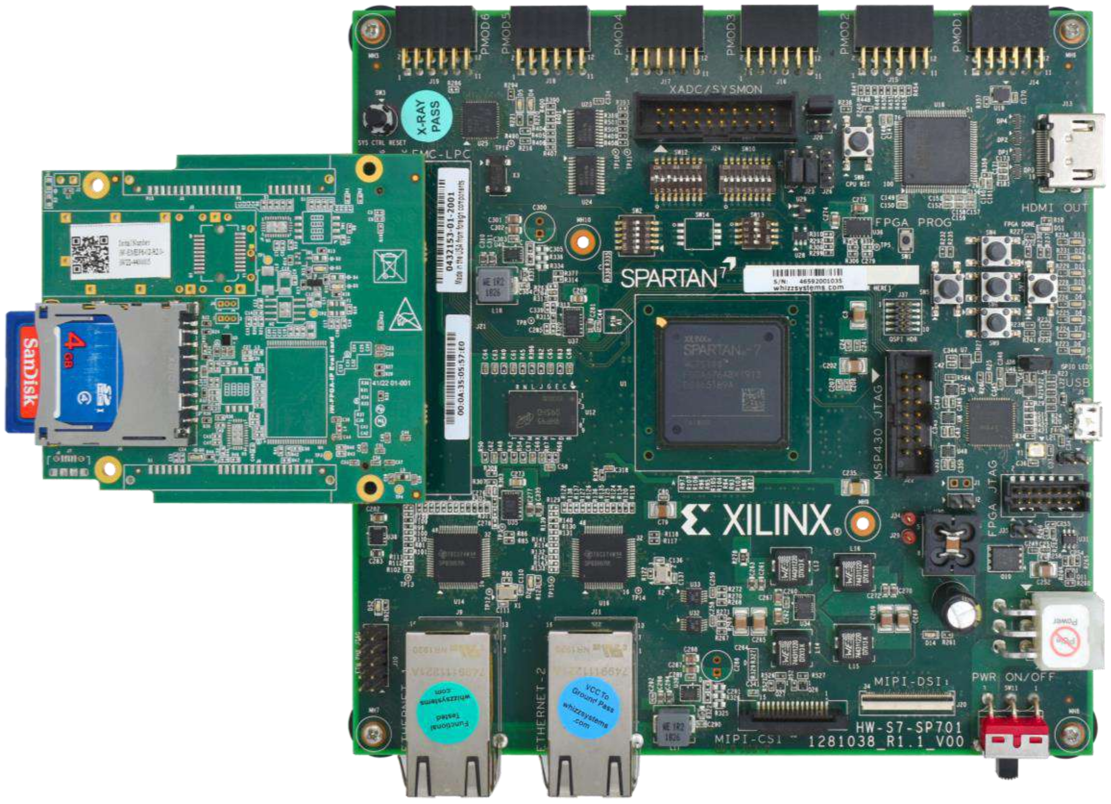


Figure 4-1:Test Setup of SP701 dev. kit with iW_FMC card

1. Connect the 12V,5A power supply
2. Power on the board
3. Make sure that iWave FMC daughter card is set to 1.8V and 2 Blue LED in FMC card is glowing as shown in the Figure Below

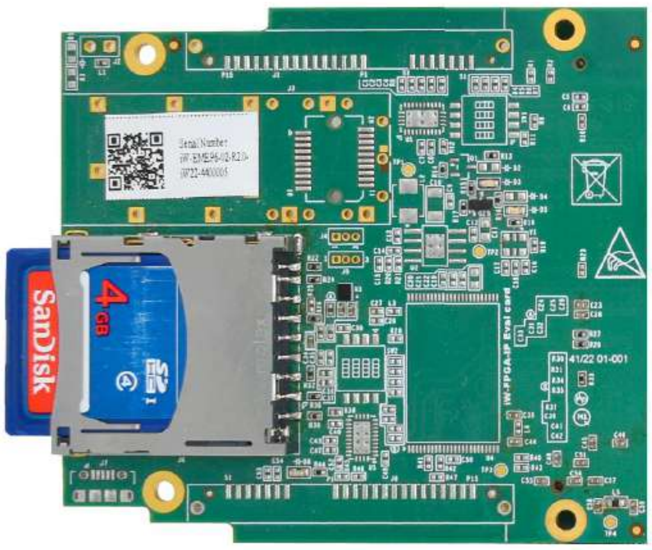
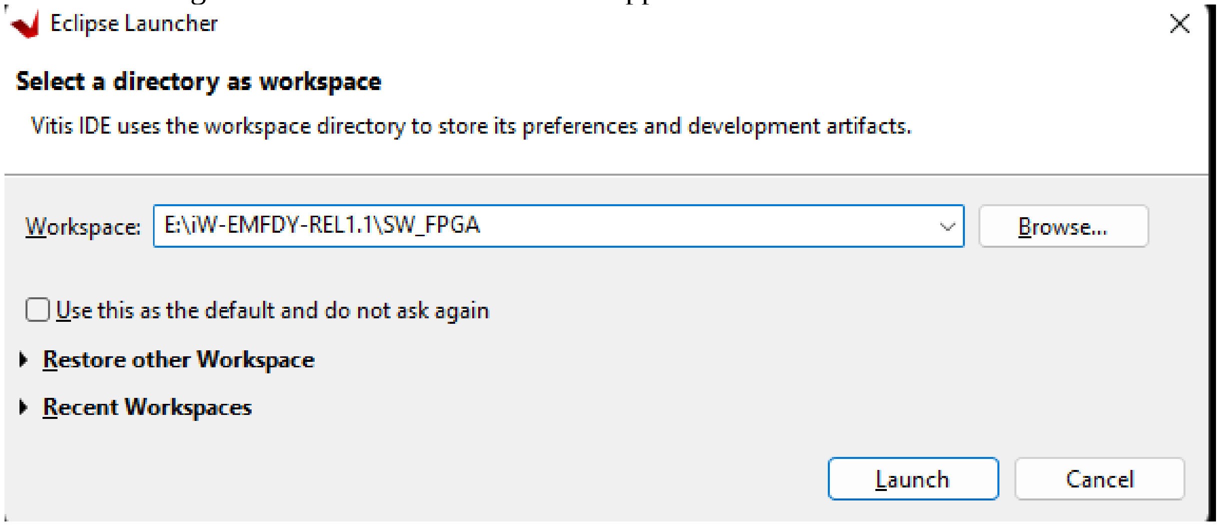


Figure 4-2:iWave FMC daughter card

Note: The iWave FMC daughter card cannot be hot-plugged.

4.3 Programming FPGA and running using Vitis.

- After generating bitstream in Vivado 2020.1 go to File→ Export hardware → select Fixed click next → select include bitstream and click next → give the XSA file name where you want to store your xsa and keep directory as local to project and click Finish
- Create a local folder in the Vivado Project folder and give that as workspace folder for Vitis 2020.1 project and click ok
- After exporting the project go to tools in Vivado and give the path of the local folder created as a workspace folder for Vitis (mentioned in step 2), click on launch Vitis IDE as shown in Figure below. A new window will appear



Eclipse Launcher

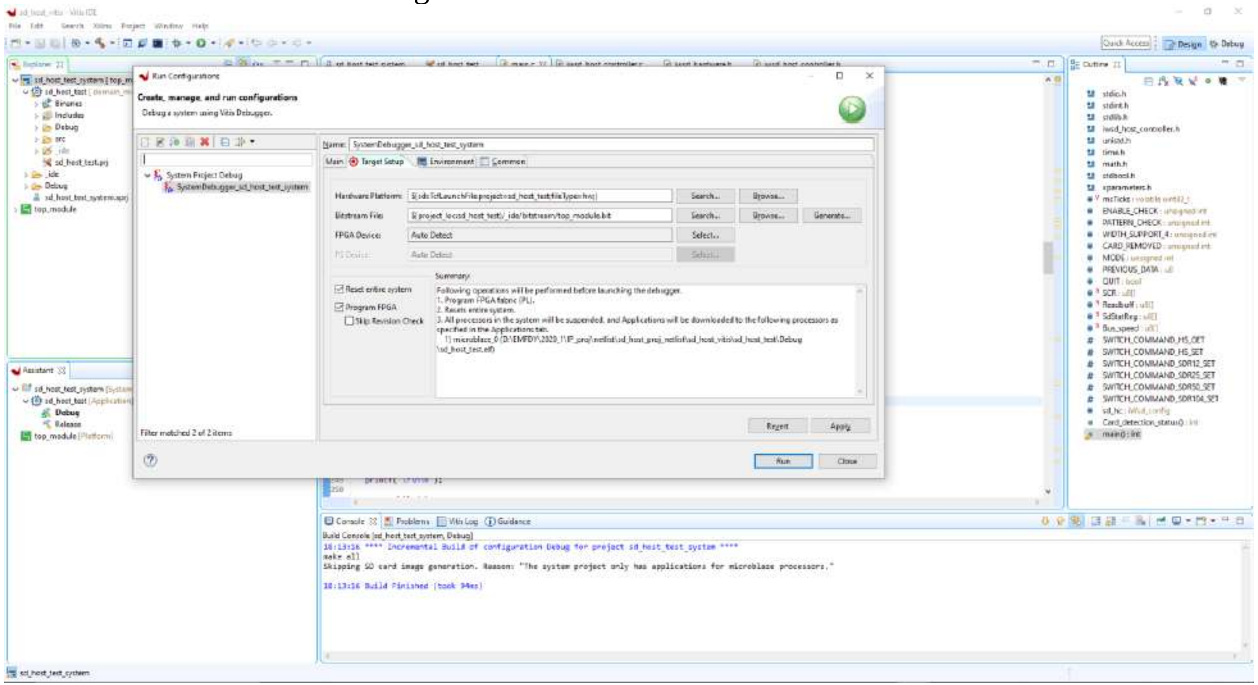
Select a directory as workspace

Vitis IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: E:\iW-EMFDY-REL1.1\SW_FPGA

Figure 4-3:Launching Vitis 2020.1

- Once Vitis is opened then the click on creates new application project and then generate the platform file giving the .xsa file path by browsing to that file in the projects local folder and click next
- In Available templets click on empty project and click on finish.
- Once the project is open, copy the source code from the release folder (..\iW-EMFDY-PF-01-R1.0-REL1.3\iW-EMFDY-SF-01-R1.0-REL1.2\iW-EMFDY-SC-01-R1.0-REL1.2\sd_host_vitis_190MHz) to Vitis project and then Save(CTRL+S) and Build (CTRL+B) the project
  NOTE: Copy only iwsd_hardware.h, iwsd_host_controller.h, iwsd_host_controller.c and main.c from the release folder to the src folder present under the application project.
- Then right click on project go to run and select run configuration system project debug
- In the appeared window under target setup click on reset entire system and program FPGA as shown in the figure below.


Run Configurations

Create, manage, and run configurations

Debug a system using Vitis Debugger.

Name: SystemDebugger_sd_host_test_system

Main | Target Setup | Environment | Common

Hardware Platform: ${sdx:TclLaunchFile:project=sd_host_test;fileType=hw}

Bitstream File: ${project_loc:sd_host_test}/_ide/bitstream/top_module.bit

FPGA Device: Auto Detect

PS Device: Auto Detect

[✓] Reset entire system

[✓] Program FPGA

[ ] Skip Revision Check

Summary:

Following operations will be performed before launching the debugger.
1. Program FPGA fabric (PL).
2. Resets entire system.
3. All processors in the system will be suspended, and Applications will be downloaded to the following processors as specified in the Applications tab.
   1) microblaze_0 (D:\EMFDY\2020_1\IP_proj\netlist\sd_host_proj_netlist\sd_host_vitis\sd_host_test\Debug\sd_host_test.elf)

Filter matched 2 of 2 items

Build Console [sd_host_test_system, Debug]

```
18:13:16 **** Incremental Build of configuration Debug for project sd_host_test_system ****
make all
Skipping SD card image generation. Reason: "The system project only has applications for microblaze processors."

18:13:16 Build Finished (took 94ms)
```

Figure 4-4:Target setup in run configuration.

- Now click on Run
- After clicking on Run FPGA will be programmed and board is test ready.

NOTE: Before clicking on Run please make sure board is turned on with JTAG connect and proper tera term setup and if program FPGA is done using Vivado please don't select program FPGA option in target setup.

5 Test Procedure

5.1 Tera term setup

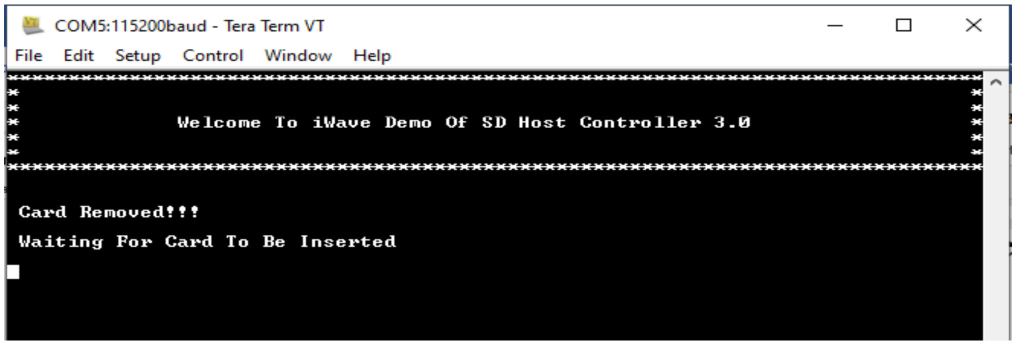
- Open tera term in PC and select the com port.
- Select baud rate as 115200.

5.2 Testing Procedure

After set up follow the below procedure to test the SD Host controller IP.

5.2.1 Initial prints:

After clicking on Run in Run configuration software will program the FPGA. As soon as the programming is done and .elf file is executed, then below print will be displayed in Tera term.


COM5:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
*****************************************************************************
*                                                                           *
*                                                                           *
*             Welcome To iWave Demo Of SD Host Controller 3.0               *
*                                                                           *
*                                                                           *
*****************************************************************************

Card Removed!!!
Waiting For Card To Be Inserted
```

Figure 5-1:Initial print after FPGA is programmed

As shown in the figure above the application is waiting for the card to be Inserted. Insert the card and then the check that the D5 Red led on the FMC card is glowing as shown in the figure below which indicates that SD card is detected,

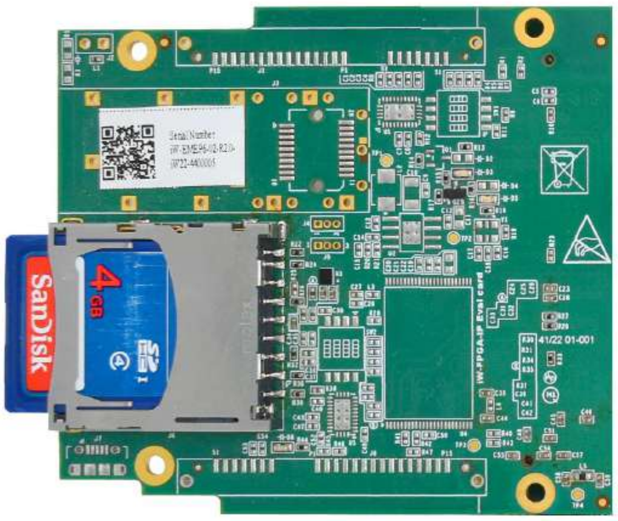
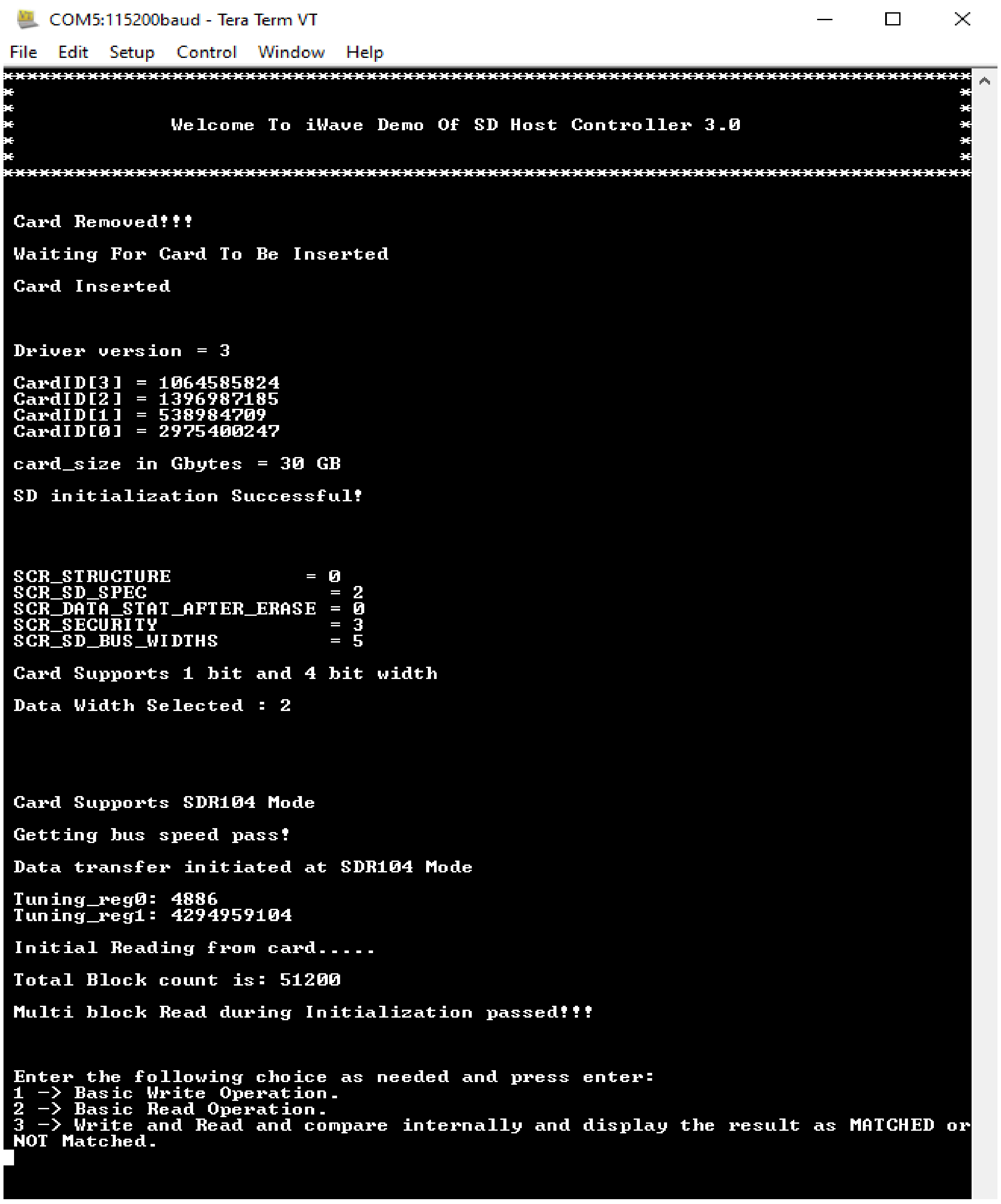


Figure 5-2:Card Inserted to FMC


COM5:115200baud - Tera Term VT
File Edit Setup Control Window Help

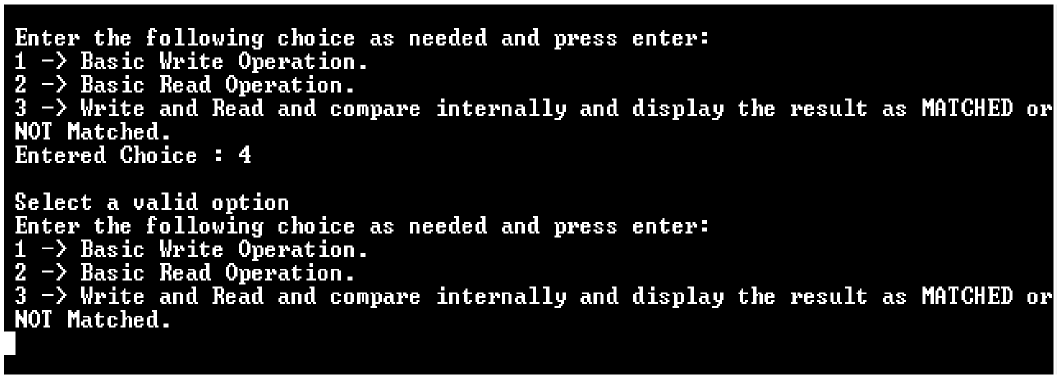
```
****************************************************************************
*                                                                          *
*                                                                          *
*           Welcome To iWave Demo Of SD Host Controller 3.0                *
*                                                                          *
*                                                                          *
****************************************************************************


Card Removed!!!

Waiting For Card To Be Inserted

Card Inserted


Driver version = 3

CardID[3] = 1064585824
CardID[2] = 1396987185
CardID[1] = 538984709
CardID[0] = 2975400247

card_size in Gbytes = 30 GB

SD initialization Successful!



SCR_STRUCTURE             = 0
SCR_SD_SPEC                 = 2
SCR_DATA_STAT_AFTER_ERASE = 0
SCR_SECURITY                = 3
SCR_SD_BUS_WIDTHS           = 5

Card Supports 1 bit and 4 bit width

Data Width Selected : 2




Card Supports SDR104 Mode

Getting bus speed pass!

Data transfer initiated at SDR104 Mode

Tuning_reg0: 4886
Tuning_reg1: 4294959104

Initial Reading from card.....

Total Block count is: 51200

Multi block Read during Initialization passed!!!


Enter the following choice as needed and press enter:
1 -> Basic Write Operation.
2 -> Basic Read Operation.
3 -> Write and Read and compare internally and display the result as MATCHED or
NOT Matched.
```

Figure 5-3:Card Initialization

As shown in the figure above 3 test case is available i.e.

1. Basic Write Operation
2. Basic Read Operation
3. Write and Read and compare internally and display the result as Matched or Not Matched

Any other test case than the mentioned above will result in wrong choice as shown in figure below.

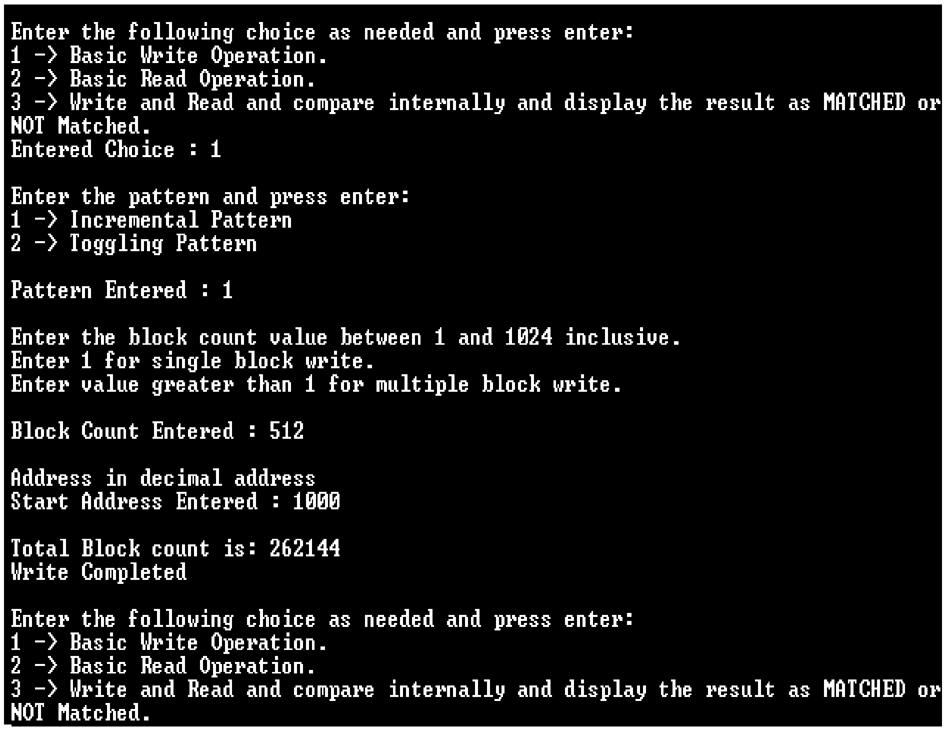


```
Enter the following choice as needed and press enter:
1 -> Basic Write Operation.
2 -> Basic Read Operation.
3 -> Write and Read and compare internally and display the result as MATCHED or
NOT Matched.
Entered Choice : 4

Select a valid option
Enter the following choice as needed and press enter:
1 -> Basic Write Operation.
2 -> Basic Read Operation.
3 -> Write and Read and compare internally and display the result as MATCHED or
NOT Matched.
```

Figure 5-4:Wrong choice

5.2.2 Basic Write Operation:

Basic write operation is the 1st testcase in the current design. After initial prints, press "**1**" and then enter the pattern enter "**1**" for **Incremental** pattern and enter "**2**" for **Toggling** pattern. Once pattern is selected then enter the block count with minimum value of "**1**" and maximum value of "**1024**" enter the block count and the enter the **Start address**, once the operation is completed successfully then user get the print "**Write Completed**."

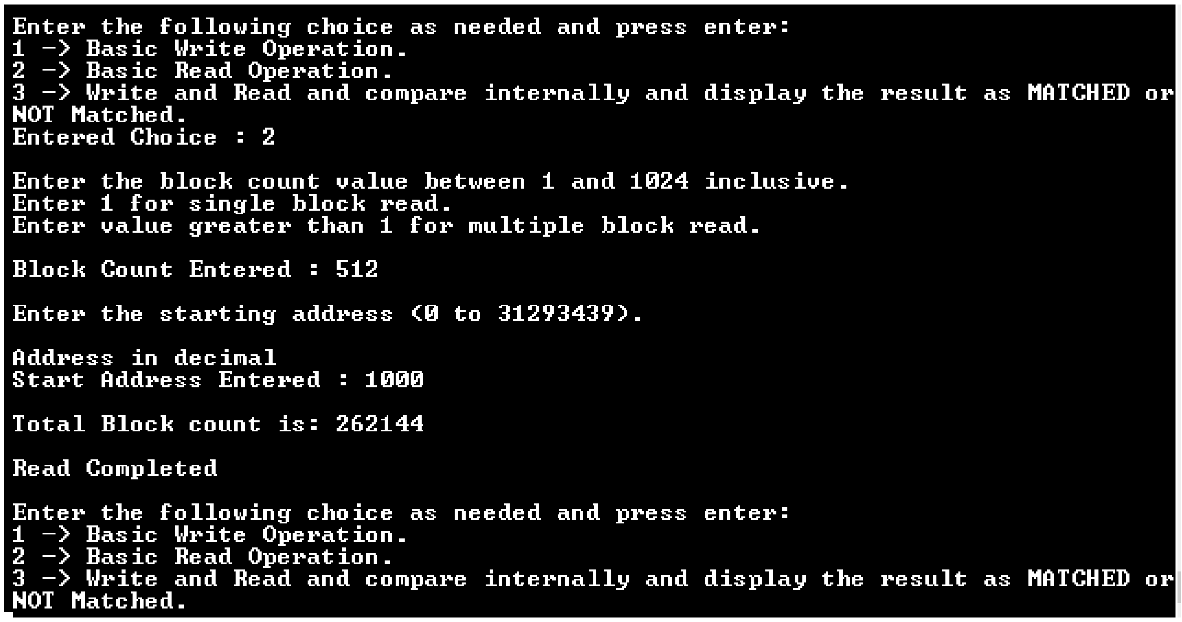


```
Enter the following choice as needed and press enter:
1 -> Basic Write Operation.
2 -> Basic Read Operation.
3 -> Write and Read and compare internally and display the result as MATCHED or
NOT Matched.
Entered Choice : 1

Enter the pattern and press enter:
1 -> Incremental Pattern
2 -> Toggling Pattern

Pattern Entered : 1

Enter the block count value between 1 and 1024 inclusive.
Enter 1 for single block write.
Enter value greater than 1 for multiple block write.

Block Count Entered : 512

Address in decimal address
Start Address Entered : 1000

Total Block count is: 262144
Write Completed

Enter the following choice as needed and press enter:
1 -> Basic Write Operation.
2 -> Basic Read Operation.
3 -> Write and Read and compare internally and display the result as MATCHED or
NOT Matched.
```

Figure 5-5:Basic Write Operation

5.2.3 Basic Read Operation:

After initial prints, press "**2**" for basic Read Operation and then enter the block count with minimum value of "**1**" and maximum value of "**1024**" enter the block count and the enter the **Start address** from where the read operation need to started, once the read operation is completed successfully then user get the print "**Read Completed**."

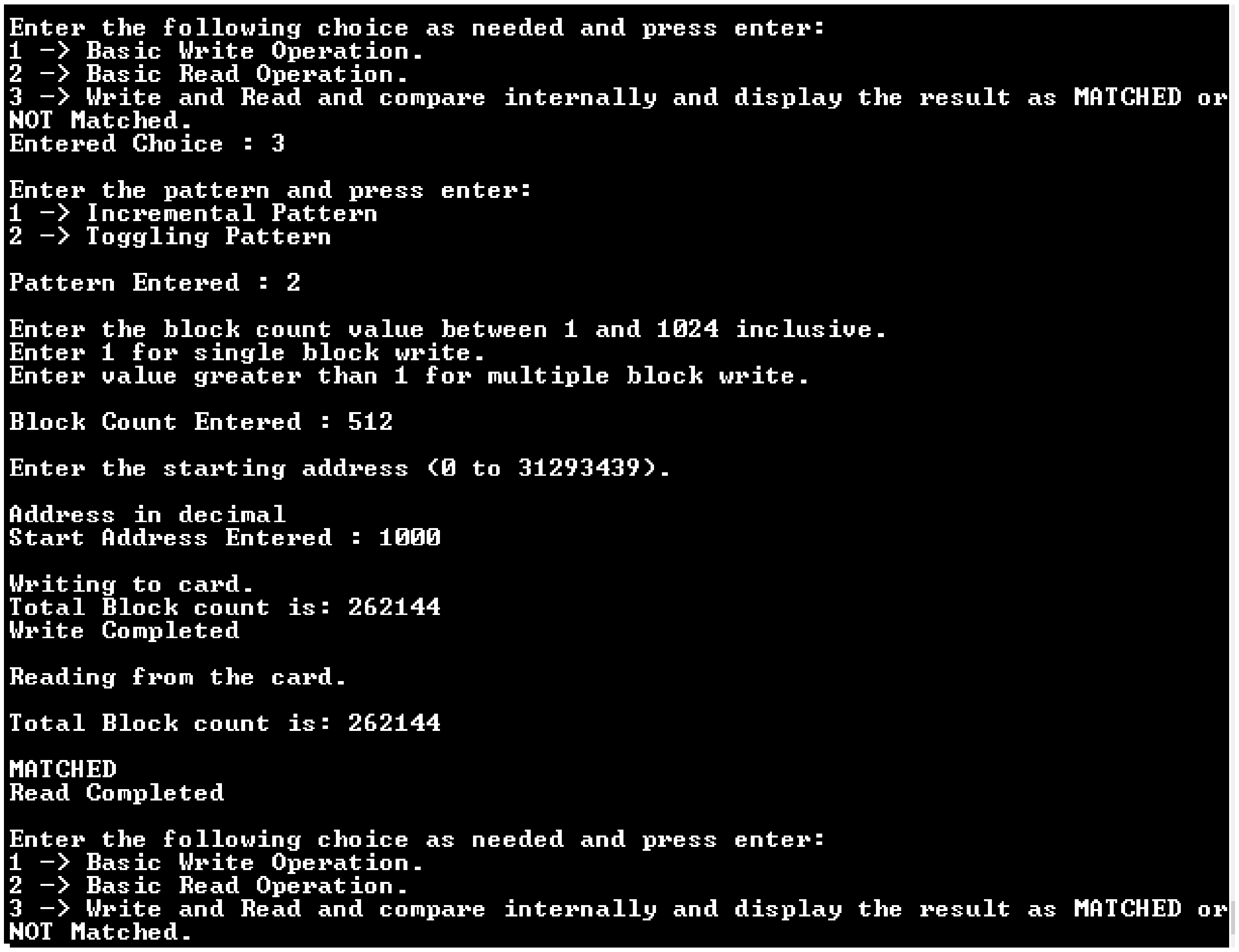


```
Enter the following choice as needed and press enter:
1 -> Basic Write Operation.
2 -> Basic Read Operation.
3 -> Write and Read and compare internally and display the result as MATCHED or
NOT Matched.
Entered Choice : 2

Enter the block count value between 1 and 1024 inclusive.
Enter 1 for single block read.
Enter value greater than 1 for multiple block read.

Block Count Entered : 512

Enter the starting address (0 to 31293439).

Address in decimal
Start Address Entered : 1000

Total Block count is: 262144

Read Completed

Enter the following choice as needed and press enter:
1 -> Basic Write Operation.
2 -> Basic Read Operation.
3 -> Write and Read and compare internally and display the result as MATCHED or
NOT Matched.
```

Figure 5-6:Basic Read Operation

5.2.4 Write , Read and Compare operation:

The 3rd test case is Write and read back and compare and display weather it's a match or not match . After initial prints, press "**3**" and then enter the pattern enter "**1**" for **Incremental** pattern and enter "**2**"for **Toggling** pattern. Once pattern is selected then enter the block count where the minimum value is "**1**" and maximum value is "**1024**" enter the block count and then enter the **Start address** of the write and read operation to start. Once the address is entered the Write to Card Operation will start and once the operation is completed successfully then user get the print "**Write Completed**," then Read from Card operation will start and once the read operation is completed successfully then user get the print of the read and write data is **Matched** or **Not Matched** and then "**Read Completed**" print will be observed in the Console



```
Enter the following choice as needed and press enter:
1 -> Basic Write Operation.
2 -> Basic Read Operation.
3 -> Write and Read and compare internally and display the result as MATCHED or
NOT Matched.
Entered Choice : 3

Enter the pattern and press enter:
1 -> Incremental Pattern
2 -> Toggling Pattern

Pattern Entered : 2

Enter the block count value between 1 and 1024 inclusive.
Enter 1 for single block write.
Enter value greater than 1 for multiple block write.

Block Count Entered : 512

Enter the starting address (0 to 31293439).

Address in decimal
Start Address Entered : 1000

Writing to card.
Total Block count is: 262144
Write Completed

Reading from the card.

Total Block count is: 262144

MATCHED
Read Completed

Enter the following choice as needed and press enter:
1 -> Basic Write Operation.
2 -> Basic Read Operation.
3 -> Write and Read and compare internally and display the result as MATCHED or
NOT Matched.
```

Figure 5-7:Write & Read Back and Compare

NOTES:

- **The current base clock is running at 200 MHz which is max clock that can be used with SP701 dev kit.**

6 Design modification to be done for Custom Board

- Update the FPGA part number according to the FPGA device available in the custom board
- Update the complete design for the selected FPGA device
- Update the pin constraints for SD interface, clock, reset and UART pins
- Update the clock constraint according to the input clock frequency
- Recompile the design to generate the new binaries and use the XSA file to create the new application project

6.1 Issues Observed when porting to different Hardware Platform

Table 2:Issues observed during the porting of design to different hardware platform

SL.NO	Observation	Reason	Solution
1	Card not responding to tuning command	Setup and Hold timing of the card may not be met while receiving CMD19	Vary the phase shift in the range of 0 to 360 between SD clock and data/command lines
2	Card failing during the write or read operation after successful tuning operation	Setup or Hold timing of the card may not be met while reading or writing the data for that particular clock routing	Vary the base clock and phase shifted clock by reducing it from 200 MHz to 150 MHz

7 Simulation

7.1 Simulation Environment

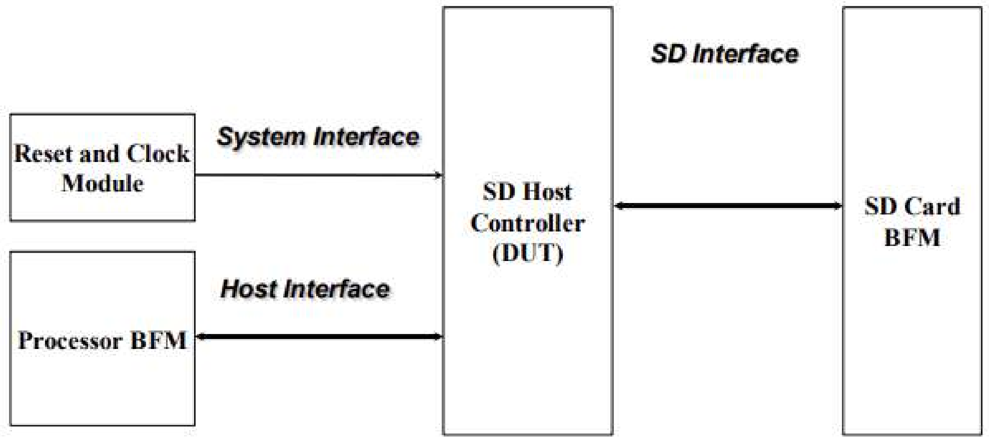


Figure 7-1: Simulation Environment Block Diagram

7.2 Simulation Environment Description

The Test Environment consists of the SD Host Controller(DUT), Processor BFM, Reset and Clock Module, and SD Card BFM.

Reset and Clock Module generates all the clock required by DUT and the reset signal. It generates the System clock of 100MHz, SD base clock frequency 100MHz with 0 and 90 degree phase shift and 200 MHz Idealy reference clock. System reset is active low signal, which is deasserted after 100ns.

Processor BFM does the read write access to DUT, Serve the ADMA request and Interrupt Request.

The SD Card BFM will receive the command/data coming from the SD interface and transmit corresponding response/data to Host through DUT.

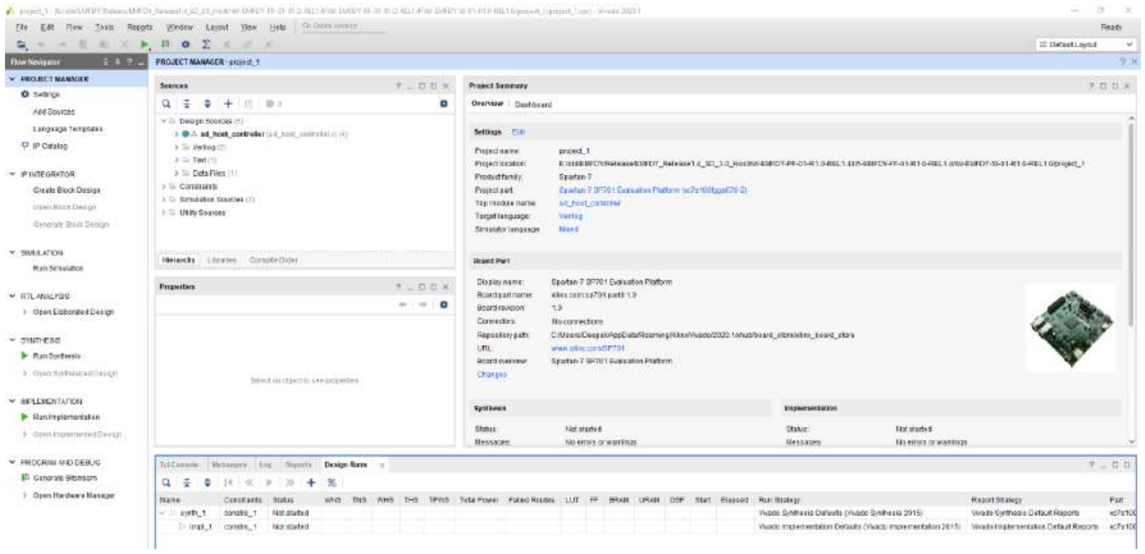
7.3 Test Case Description

The Single Test case will does the below function

- Initialization of SD card: The SD Data width mode and Speed mode configuration is done
- Single block write: 512 Byte Single Block mode write using ADMA
- Single Block Read: 512 Byte Single Block Read using ADMA
- Multi Block Write: 512 Byte Multi Block(4 Block) mode write using ADMA
- Multi Block read : 512 Byte Multi Block(4 Block) mode Read using ADMA

7.4 Step to Run Simulation

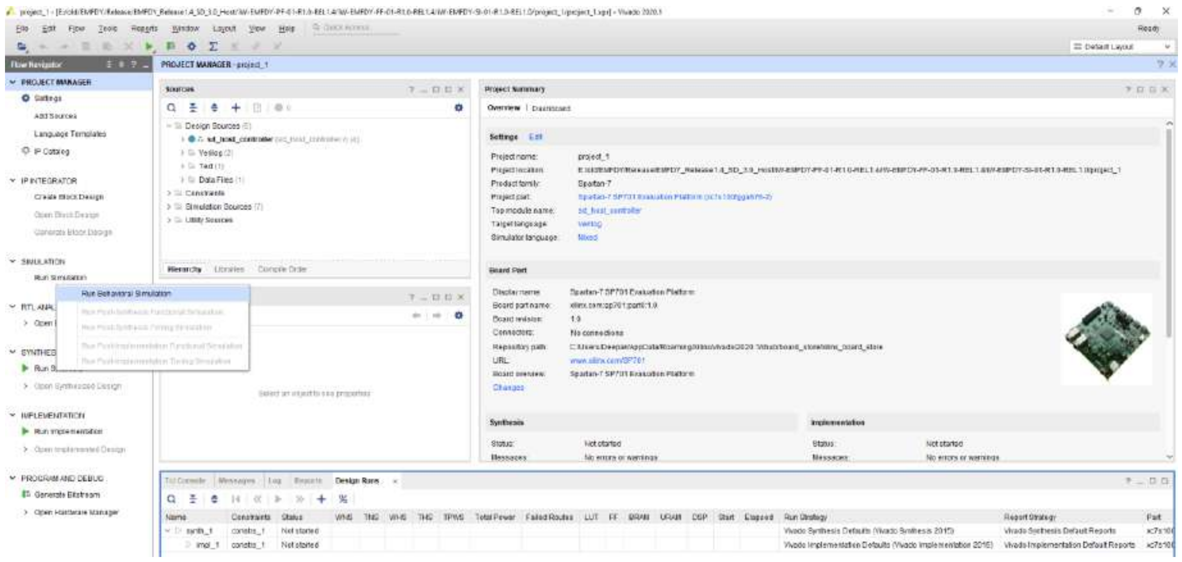
1. Open the project in the path **../EMFDY_Release1.4_SD_3.0_Host\iW-EMFDY-PF-01-R1.0-REL1.4\iW-EMFDY-FF-01-R1.0-REL1.4\iW-EMFDY-SI-01-R1.0-REL1.0/project_1** using Vivado 2020.1 as shown below



Project name:	project_1
Product family:	Spartan-7
Project part:	Spartan-7 SP701 Evaluation Platform (xc7s100fgga676-2)
Top module name:	sd_host_controller
Target language:	Verilog
Simulator language:	Mixed
Display name:	Spartan-7 SP701 Evaluation Platform
Board part name:	xilinx.com:sp701:part0:1.0
Board revision:	1.0
Connectors:	No connections
URL:	www.xilinx.com/SP701
Board overview:	Spartan-7 SP701 Evaluation Platform
Status:	Not started
Messages:	No errors or warnings

Figure 7-2: Simulation step1

2. In Simulation Tab Select Run Simulation -> Run Behavioral Simulation as shown below



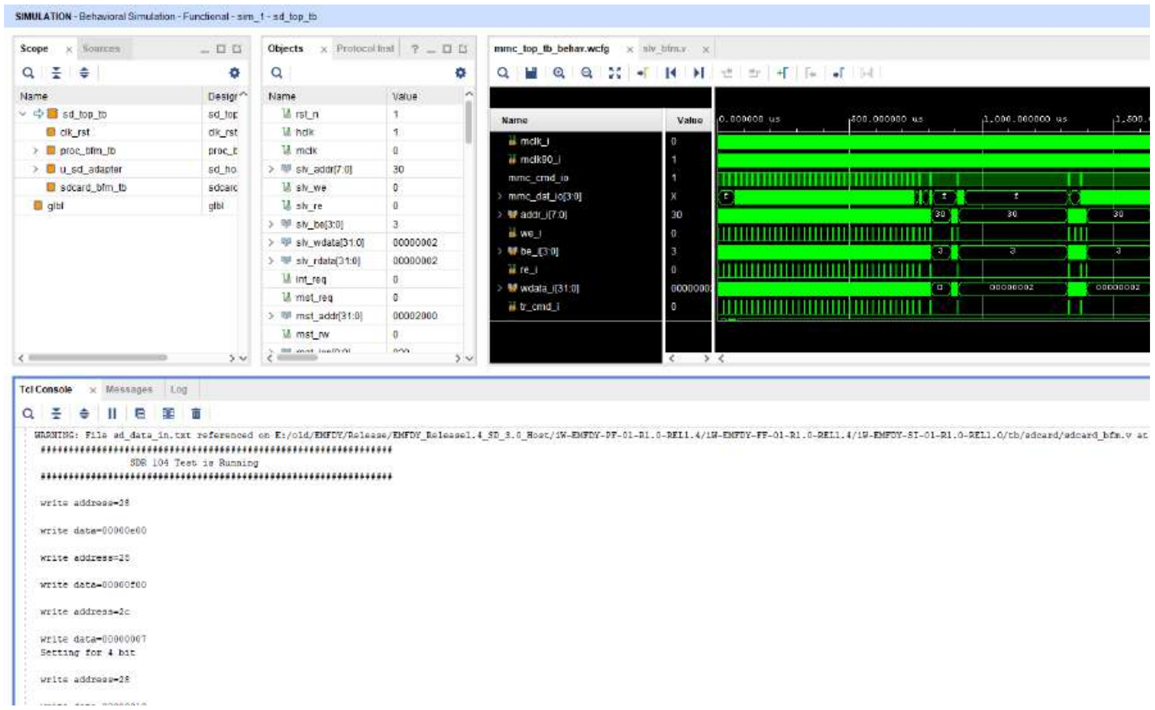
SIMULATION
Run Simulation
Run Behavioral Simulation
Run Post-Synthesis Functional Simulation
Run Post-Synthesis Timing Simulation
Run Post-Implementation Functional Simulation
Run Post-Implementation Timing Simulation

Figure 7-3: Simulation step2

3. Press Ok of unable to open the file window pops up

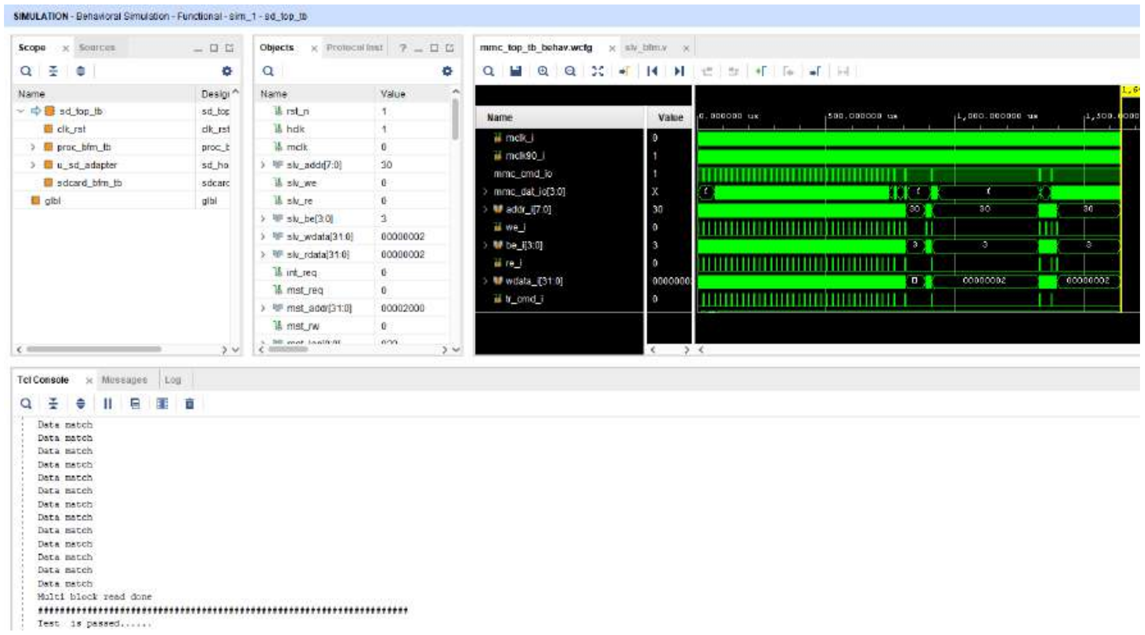
7.5 Simulation PASS Criterion

Below prints should come in the TCL console showing the start and end of the test case. User can check the TCL console prints for low level details on simulation steps which includes the register access, commands transmission, response reception and data integrity status.



```
WARNING: File sd_data_in.txt referenced on E:/old/EMFDY/Release/EMFDY_Release1.4_SD_3.0_Host/iW-EMFDY-PF-01-R1.0-REL1.4/iW-EMFDY-FF-01-R1.0-REL1.4/iW-EMFDY-SI-01-R1.0-REL1.0/tb/sdcard/sdcard_bfm.v at
 ##############################################################
                SDR 104 Test is Running
 ##############################################################

 write address=28

 write data=00000e00

 write address=28

 write data=00000f00

 write address=2c

 write data=00000007
 Setting for 4 bit

 write address=28
```

Figure 7-4: Test result print 1



```
 Data match
 Data match
 Data match
 Data match
 Data match
 Data match
 Data match
 Data match
 Data match
 Data match
 Data match
 Data match
 Data match
 Multi block read done
 ######################################################################
 Test  is passed......
```

Figure 7-5: Test result print 2

8 Resource Utilization

The table below shows the resource utilization summary for SP701 dev. kit. for SD Host Controller IP.

Table 3: Resource Utilization for device for SP701 dev. kit .

Resources	Utilization	Available
LUTs	1036	64000
LUTRAM	0	17600
FF	1277	128000
BRAM	1	120