SD Memory Slave Controller IP
User Manual

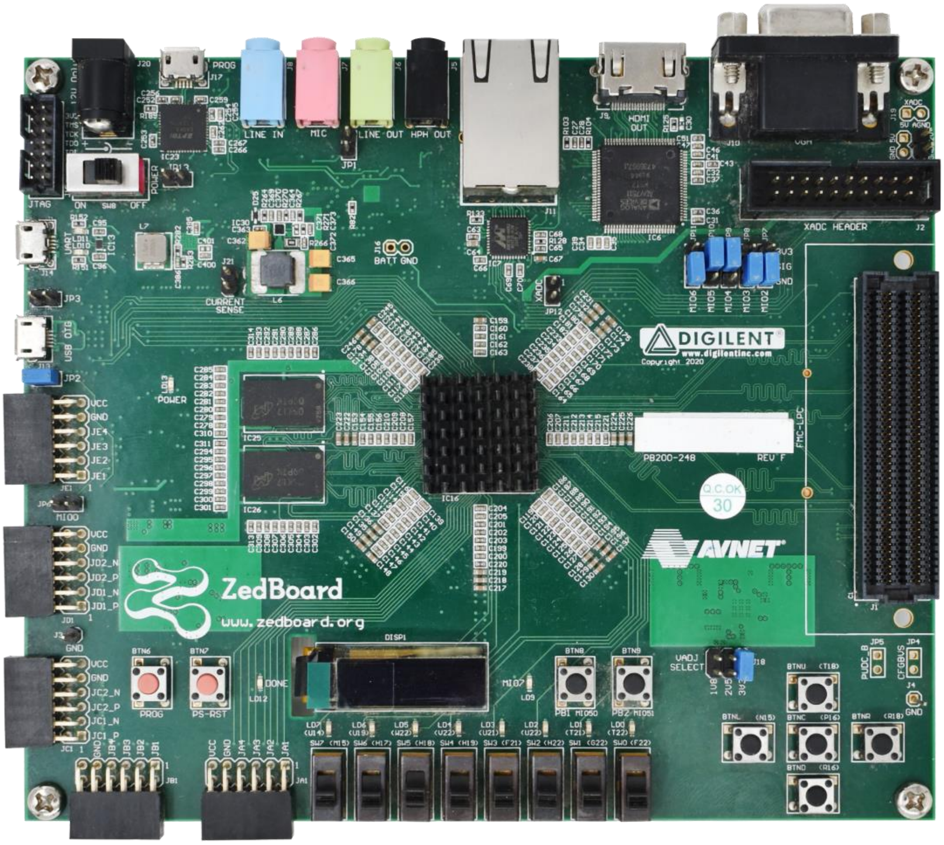


Table of Content

List Of Figures

List Of Tables

1. Introduction

The purpose of this document is to explain the procedure to power-on and setting up the working environment of iW-SD Memory Slave IP using any Host such as i.MX dev. Kit, Windows Laptop and Linux Laptop.

1.1 Scope

This document describes the Hardware connection procedure to power-on, inserting SD Memory slave demo board. This document also describes the steps to perform the write and read tests to verify the working of iW-SD Memory slave IP.

1.2 Acronyms and Abbreviations

Term	Meaning
FPGA	Field Programmable Gate Array
JTAG	Joint Test Attachment Group
MMC	Multi-Media Card
OS	Operating System
PC	Personal Computer
SD	Secure Digital
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

Table 1: Acronyms & Abbreviations

1.3 Overview

The iMX6 Qseven Development Kit ,Windows Laptop and Linux Laptop are used to evaluate SD Slave controller IP. It has SD/MMC slot in it by which we can test iW-SD memory slave IP by connecting the SD extension cable.

1.4 Board and accessories

- Zed Board
- i.MX6 Qseven Development kit , Windows Laptop and Linux Laptop.
- SD extension cable
- 2 USB cable 1 each for UART and JTAG.

1.5 Software Requirements

- A Laptop/PC with tera term installed.
- Vivado 2020.1 with Vitis

2. Test setup

2.1 Connections

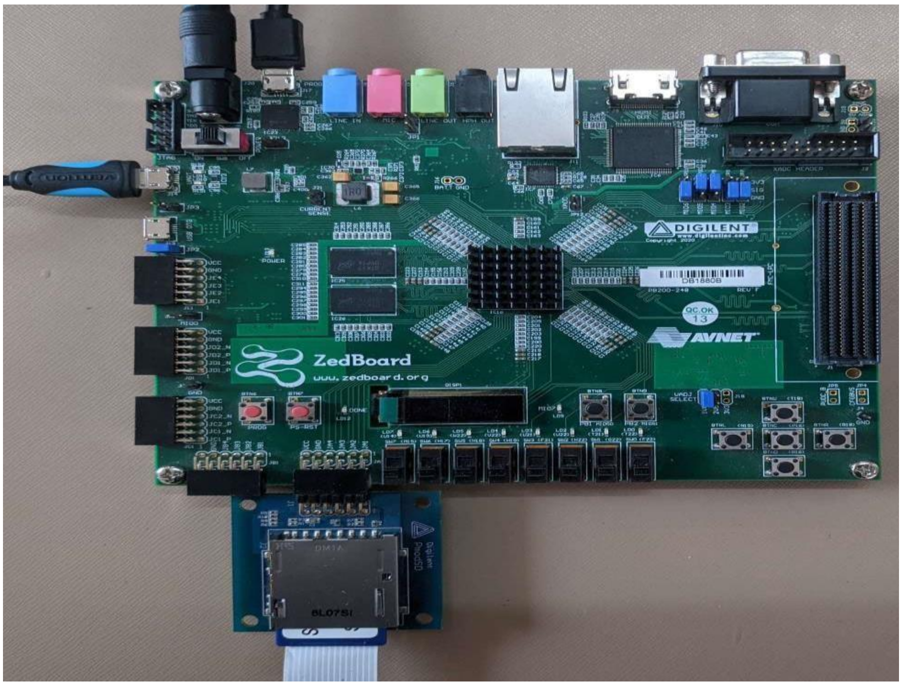


Figure 1: UART Cable and SD extension cable connection

Do the following connection as shown in the figure above.

- SD Extension cable to test IP
- 1 UART cable for Programming
- 1 UART cable for Serial Monitor

2.2 Programming FPGA and running software using Vitis

- After generating bitstream in vivado , go to export hardware

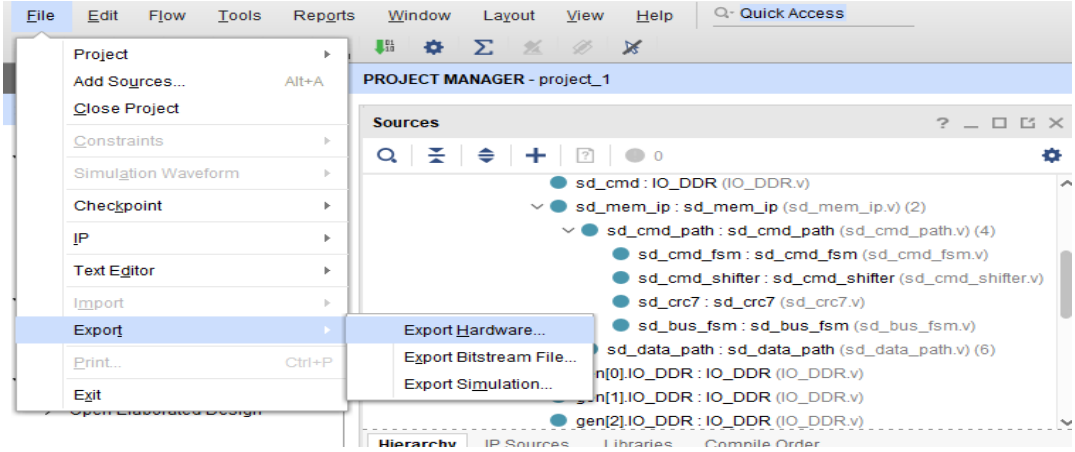
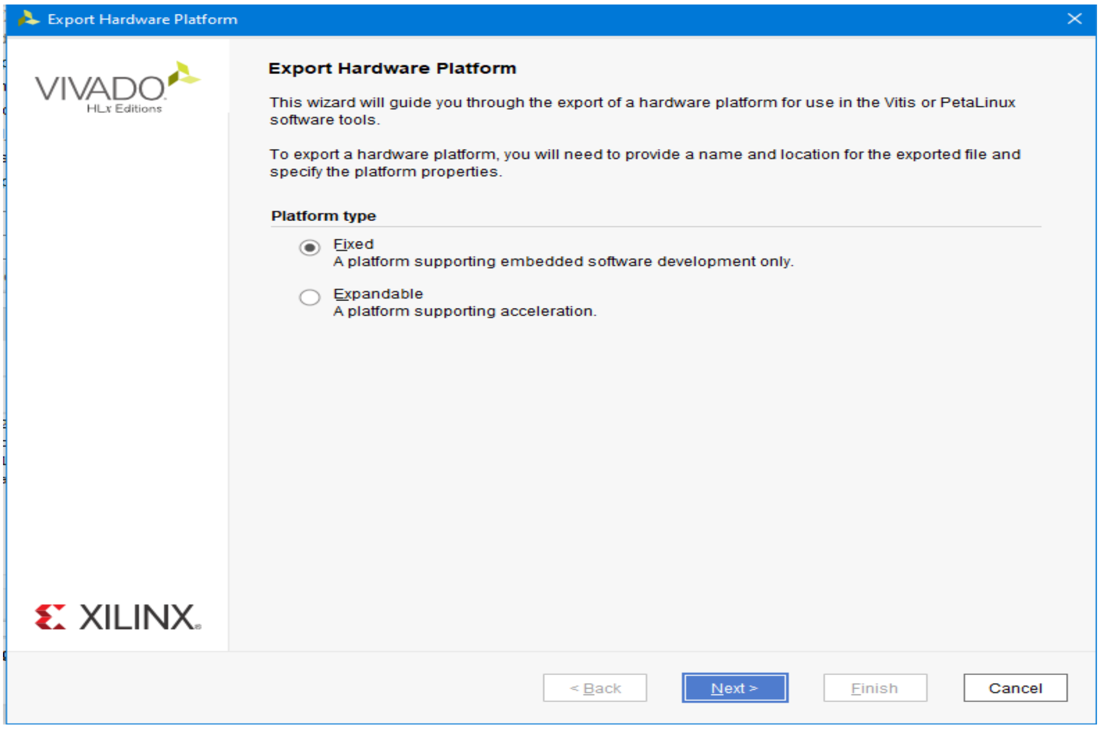

File
Edit
Flow
Tools
Reports
Window
Layout
View
Help
Quick Access
Project
Add Sources... Alt+A
Close Project
Constraints
Simulation Waveform
Checkpoint
IP
Text Editor
Import
Export
Print... Ctrl+P
Exit
Export Hardware...
Export Bitstream File...
Export Simulation...
PROJECT MANAGER - project_1
Sources
sd_cmd : IO_DDR (IO_DDR.v)
sd_mem_ip : sd_mem_ip (sd_mem_ip.v) (2)
sd_cmd_path : sd_cmd_path (sd_cmd_path.v) (4)
sd_cmd_fsm : sd_cmd_fsm (sd_cmd_fsm.v)
sd_cmd_shifter : sd_cmd_shifter (sd_cmd_shifter.v)
sd_crc7 : sd_crc7 (sd_crc7.v)
sd_bus_fsm : sd_bus_fsm (sd_bus_fsm.v)
sd_data_path : sd_data_path (sd_data_path.v) (6)
n[0].IO_DDR : IO_DDR (IO_DDR.v)
n[1].IO_DDR : IO_DDR (IO_DDR.v)
gen[2].IO_DDR : IO_DDR (IO_DDR.v)
Hierarchy
IP Sources
Libraries
Compile Order

Figure 2: Exporting Hardware from Vivado

- Choose Fixed Type Platform



Export Hardware Platform

This wizard will guide you through the export of a hardware platform for use in the Vitis or PetaLinux software tools.

To export a hardware platform, you will need to provide a name and location for the exported file and specify the platform properties.

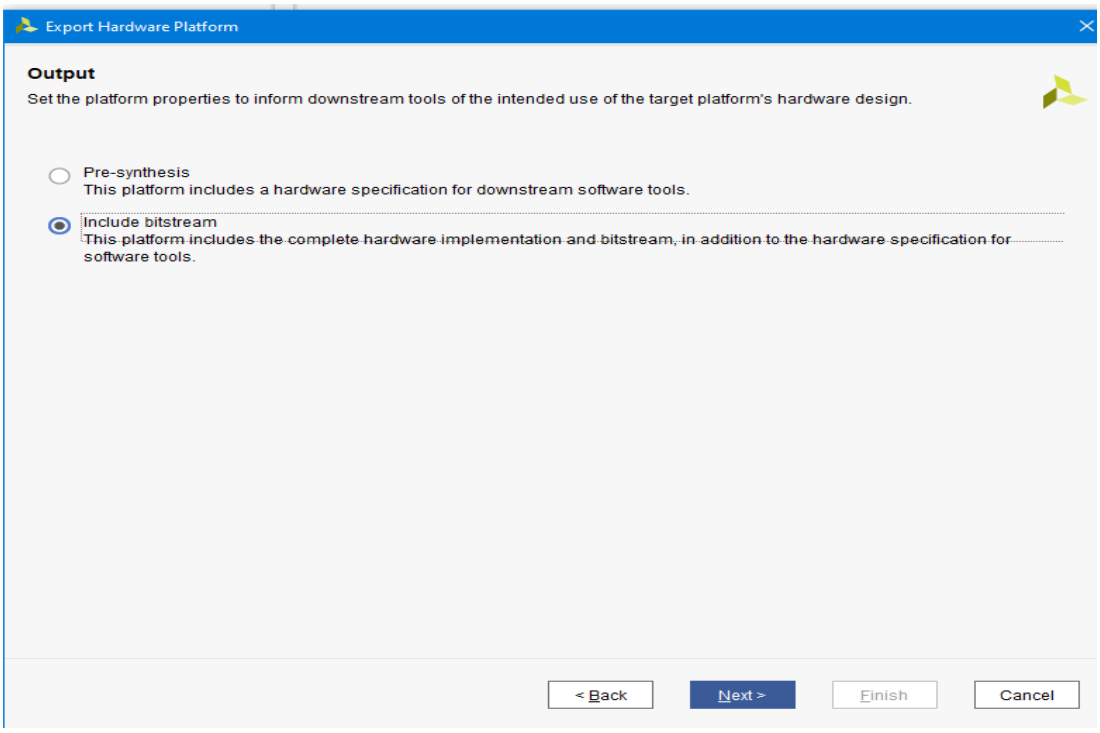
Platform type

● Fixed
A platform supporting embedded software development only.

○ Expandable
A platform supporting acceleration.

Figure 3: Choosing Fixed Platform Type

- Click to include bitstream and then click on Next



Output

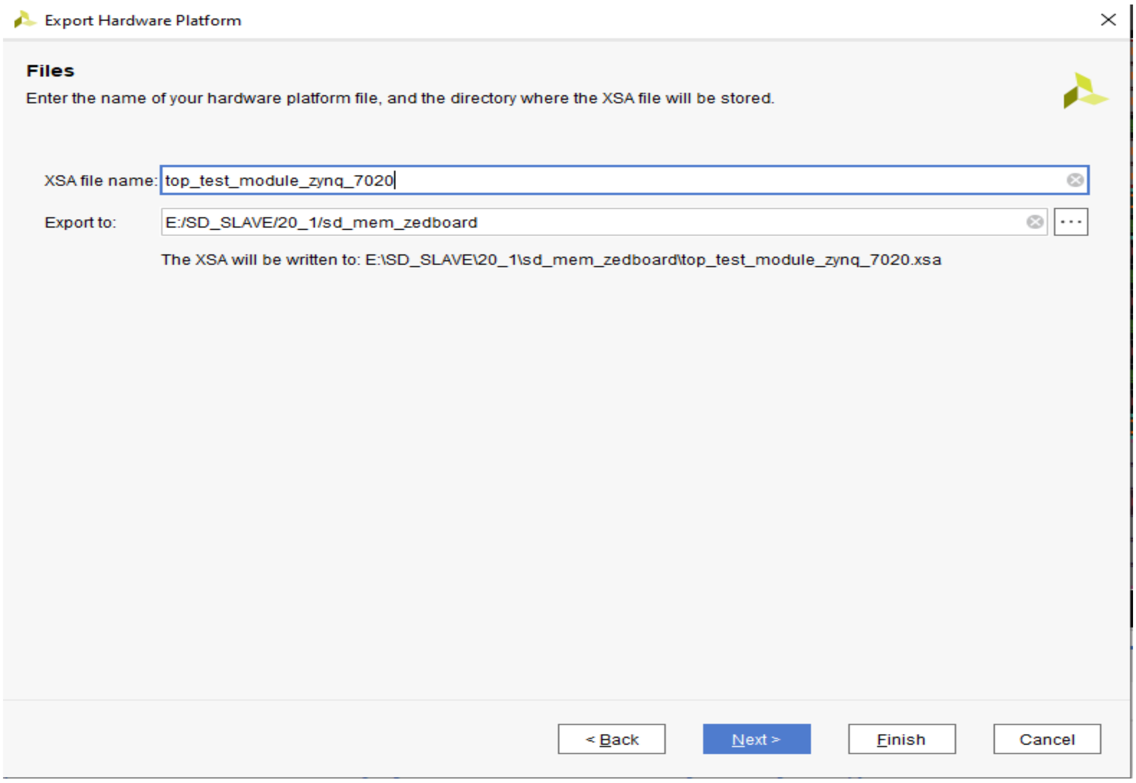
Set the platform properties to inform downstream tools of the intended use of the target platform's hardware design.

○ Pre-synthesis
This platform includes a hardware specification for downstream software tools.

● Include bitstream
This platform includes the complete hardware implementation and bitstream, in addition to the hardware specification for software tools.

Figure 4: *Including the Bitstream*

- Give name to XSA , Click Next and then Click Finish



Export Hardware Platform

Files

Enter the name of your hardware platform file, and the directory where the XSA file will be stored.

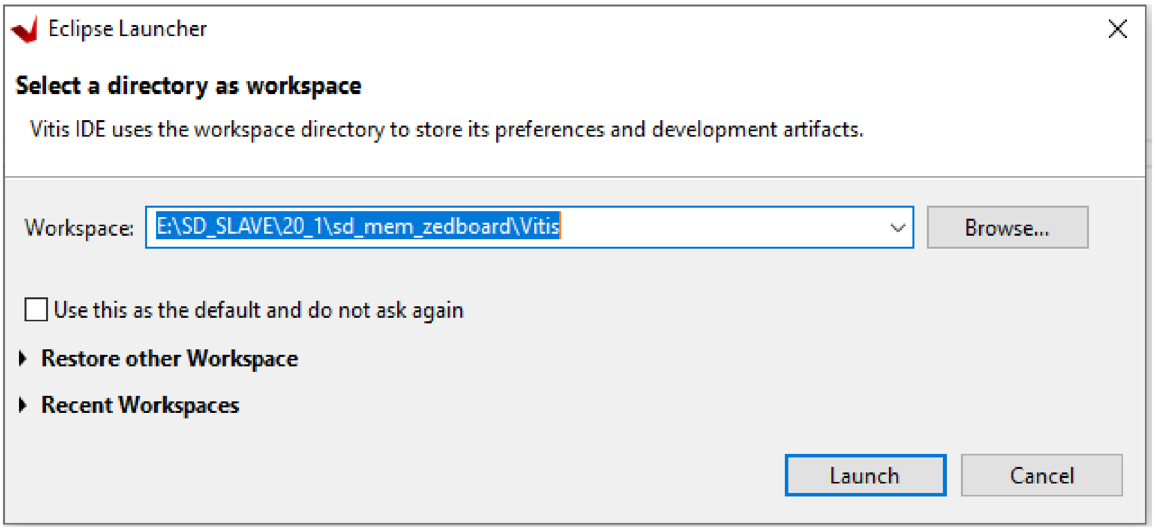
XSA file name: top_test_module_zynq_7020

Export to: E:/SD_SLAVE/20_1/sd_mem_zedboard

The XSA will be written to: E:\SD_SLAVE\20_1\sd_mem_zedboard\top_test_module_zynq_7020.xsa

Figure 5*: Naming the XSA*

- Now open Vitis and select the specified directory and press launch



Eclipse Launcher

Select a directory as workspace

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

Workspace: E:\SD_SLAVE\20_1\sd_mem_zedboard\Vitis

☐ Use this as the default and do not ask again

Figure 6: Select the Specified Project Directory

- Click Create Application Project as shown in the below Figure and then Click Next.

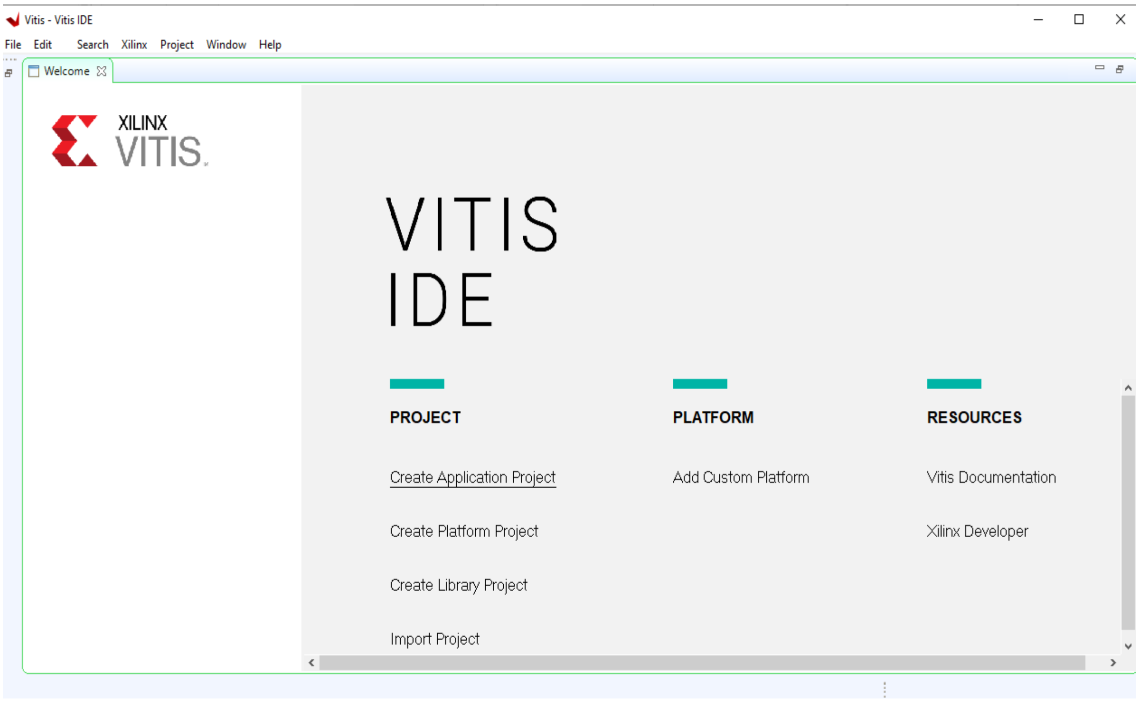
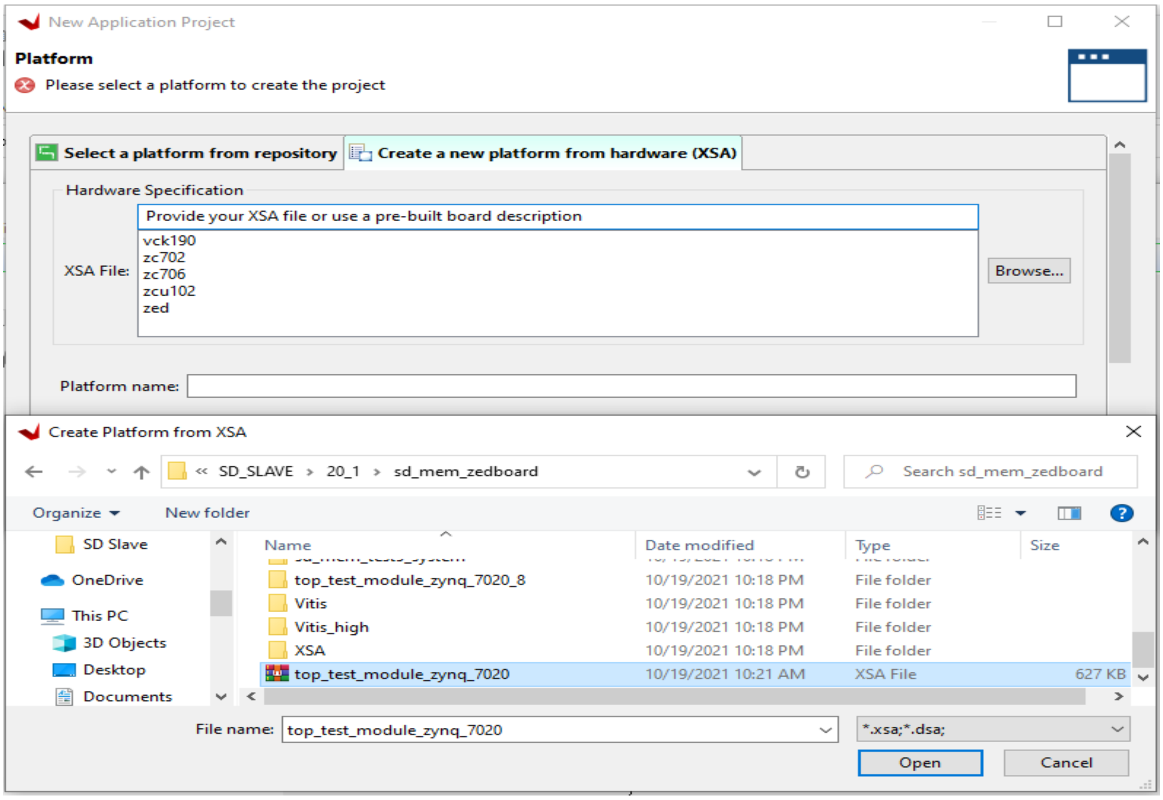


Figure 7: Create Application Project

- Select the XSA from the Project Folder



Platform

Please select a platform to create the project

Hardware Specification

XSA File: Provide your XSA file or use a pre-built board description

vck190
zc702
zc706
zcu102
zed

Platform name:

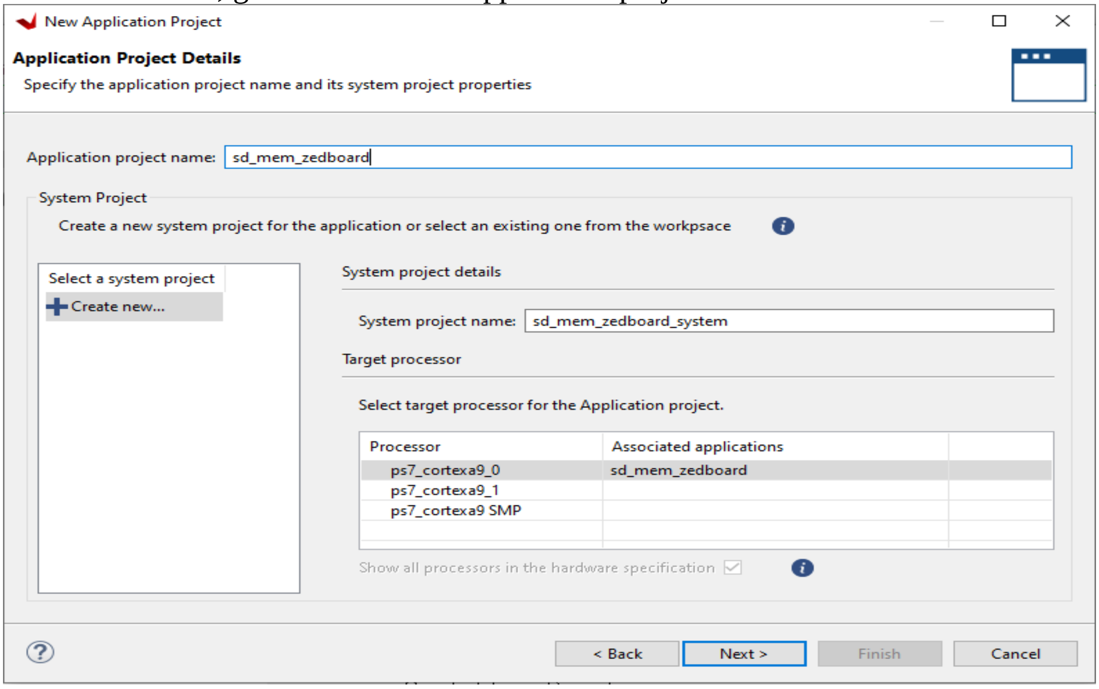
Name	Date modified	Type	Size
top_test_module_zynq_7020_8	10/19/2021 10:18 PM	File folder	
Vitis	10/19/2021 10:18 PM	File folder	
Vitis_high	10/19/2021 10:18 PM	File folder	
XSA	10/19/2021 10:18 PM	File folder	
top_test_module_zynq_7020	10/19/2021 10:21 AM	XSA File	627 KB

File name: top_test_module_zynq_7020

.xsa;.dsa;

Figure 8: Select the XSA

- Click Next , give Name to the Application project and Click Next



Application Project Details

Specify the application project name and its system project properties

Application project name: sd_mem_zedboard

System Project

Create a new system project for the application or select an existing one from the workpsace

Select a system project

Create new...

System project details

System project name: sd_mem_zedboard_system

Target processor

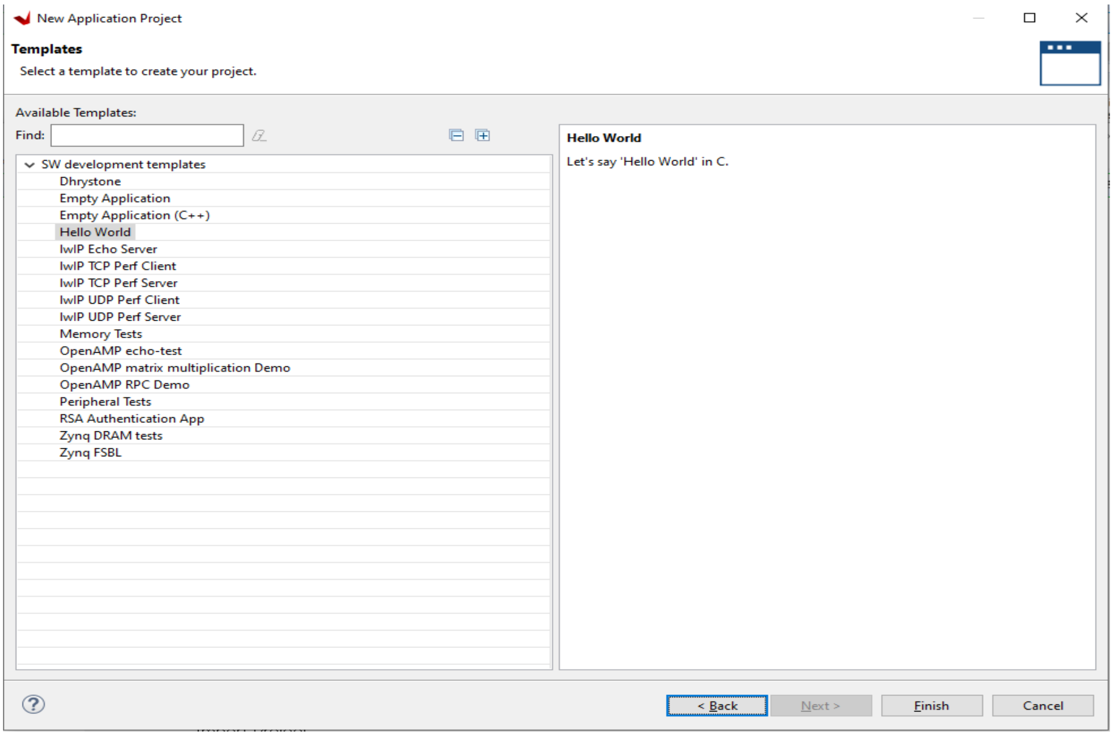
Select target processor for the Application project.

Processor	Associated applications
ps7_cortexa9_0	sd_mem_zedboard
ps7_cortexa9_1	
ps7_cortexa9 SMP	

Show all processors in the hardware specification ☑

Figure 9: Naming the Application Project

- Select hello world example template and click Finish.



Templates

Select a template to create your project.

Available Templates:

Find:

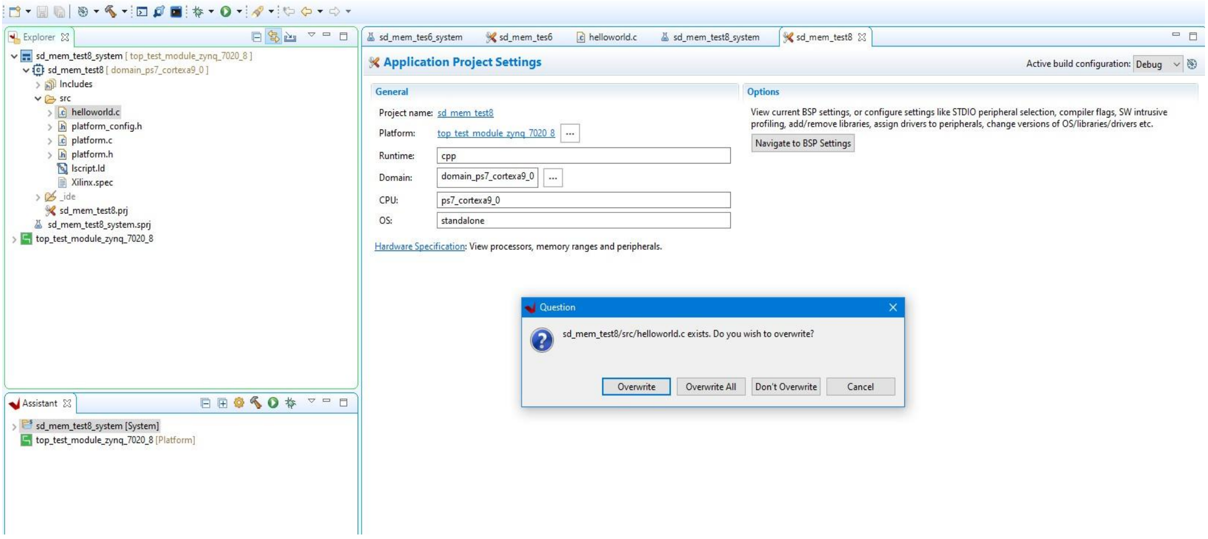
SW development templates
- Dhrystone
- Empty Application
- Empty Application (C++)
- Hello World
- lwIP Echo Server
- lwIP TCP Perf Client
- lwIP TCP Perf Server
- lwIP UDP Perf Client
- lwIP UDP Perf Server
- Memory Tests
- OpenAMP echo-test
- OpenAMP matrix multiplication Demo
- OpenAMP RPC Demo
- Peripheral Tests
- RSA Authentication App
- Zynq DRAM tests
- Zynq FSBL

Hello World

Let's say 'Hello World' in C.

Figure 10: Select Hello World Template

- Once the project is created ,replace the helloworld.c file with the given helloworld.c file.


Application Project Settings

General

Project name: sd_mem_test8
Platform: top_test_module_zynq_7020_8
Runtime: cpp
Domain: domain_ps7_cortexa9_0
CPU: ps7_cortexa9_0
OS: standalone

Hardware Specification: View processors, memory ranges and peripherals.

Options

View current BSP settings, or configure settings like STDIO peripheral selection, compiler flags, SW intrusive profiling, add/remove libraries, assign drivers to peripherals, change versions of OS/libraries/drivers etc.

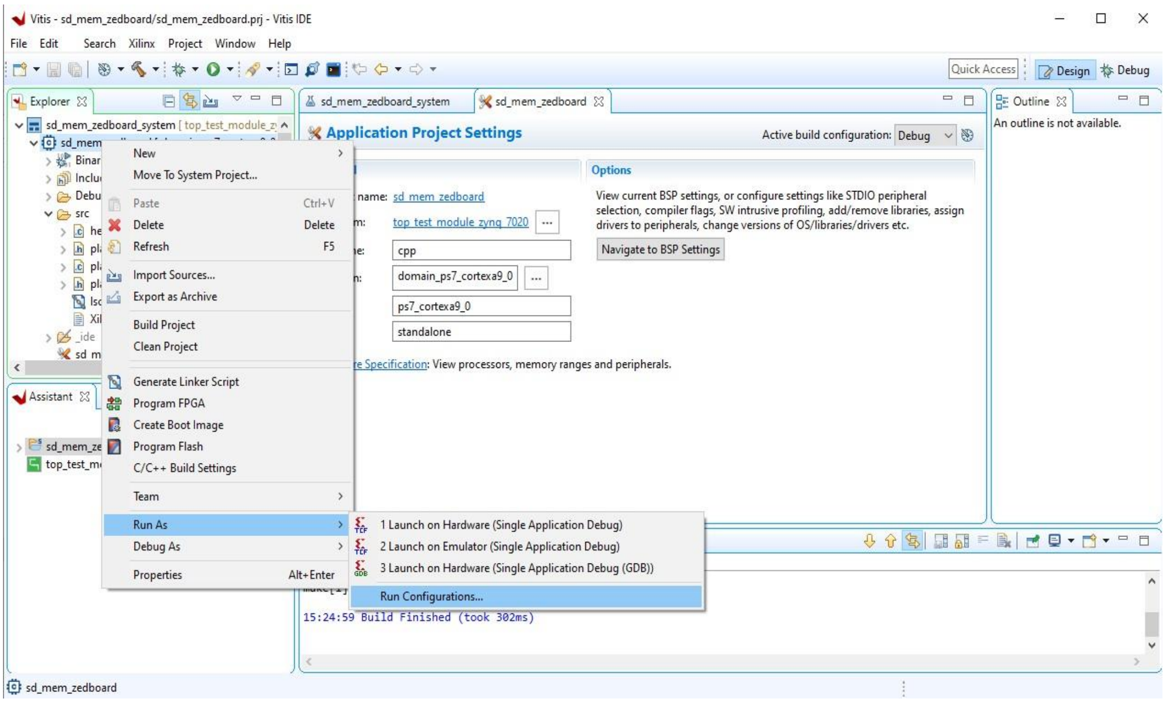
Active build configuration: Debug

Question

sd_mem_test8/src/helloworld.c exists. Do you wish to overwrite?

Figure 11: Replace Hello World File

- Save (CTRL+S) and build (CTRL+B) the project
- Then right click on project go to run and select run configuration


Vitis - sd_mem_zedboard/sd_mem_zedboard.prj - Vitis IDE

Application Project Settings

name: sd_mem_zedboard
m: top_test_module_zynq_7020
e: cpp
n: domain_ps7_cortexa9_0
ps7_cortexa9_0
standalone

e Specification: View processors, memory ranges and peripherals.

Options

View current BSP settings, or configure settings like STDIO peripheral selection, compiler flags, SW intrusive profiling, add/remove libraries, assign drivers to peripherals, change versions of OS/libraries/drivers etc.

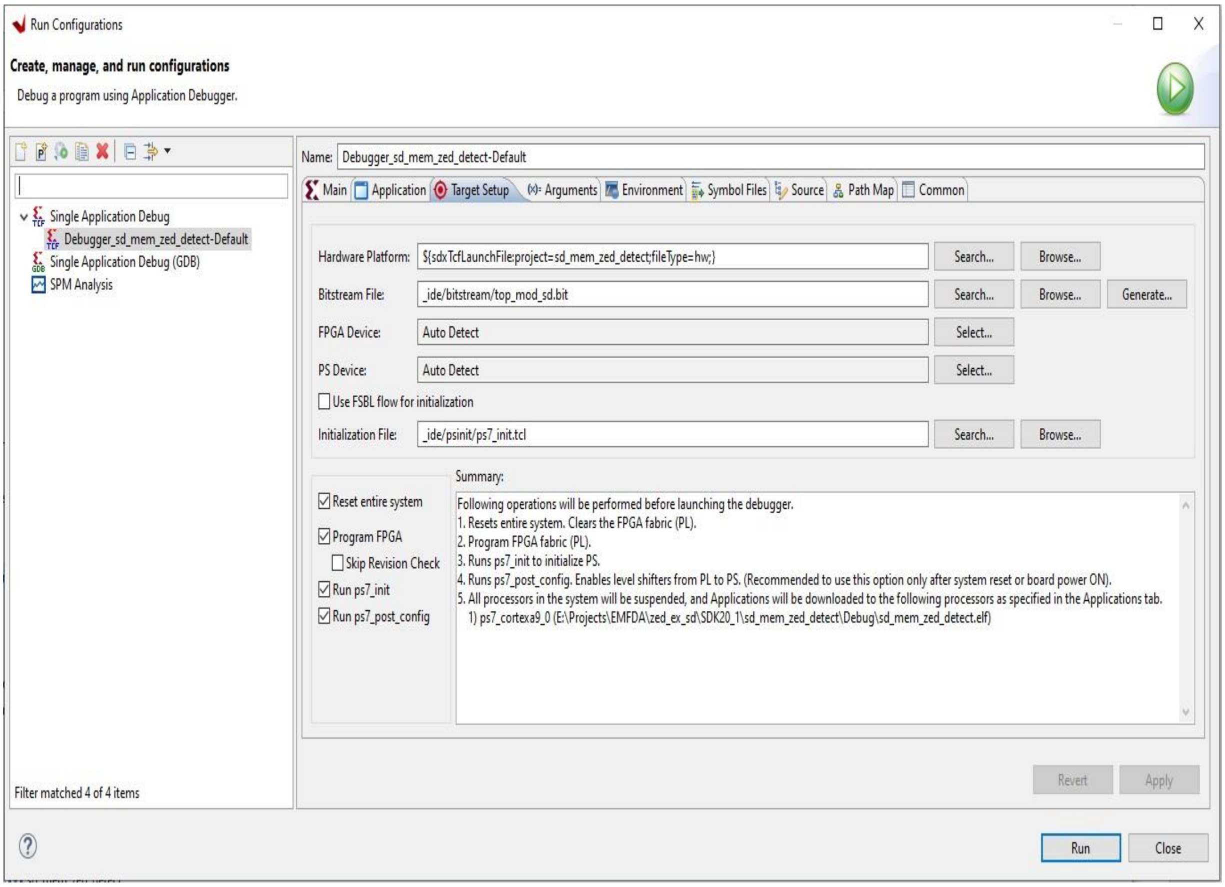
Active build configuration: Debug

An outline is not available.

```
15:24:59 Build Finished (took 302ms)
```

Figure 12: Run Configurations

- double click **single application debug** and switch to **Target Setup** tab.



Name: Debugger_sd_mem_zed_detect-Default

Hardware Platform: ${sdxTcfLaunchFile:project=sd_mem_zed_detect;fileType=hw;}

Bitstream File: _ide/bitstream/top_mod_sd.bit

FPGA Device: Auto Detect

PS Device: Auto Detect

□ Use FSBL flow for initialization

Initialization File: _ide/psinit/ps7_init.tcl

☑ Reset entire system

☑ Program FPGA

□ Skip Revision Check

☑ Run ps7_init

☑ Run ps7_post_config

Summary:

```
Following operations will be performed before launching the debugger.
1. Resets entire system. Clears the FPGA fabric (PL).
2. Program FPGA fabric (PL).
3. Runs ps7_init to initialize PS.
4. Runs ps7_post_config. Enables level shifters from PL to PS. (Recommended to use this option only after system reset or board power ON).
5. All processors in the system will be suspended, and Applications will be downloaded to the following processors as specified in the Applications tab.
   1) ps7_cortexa9_0 (E:\Projects\EMFDA\zed_ex_sd\SDK20_1\sd_mem_zed_detect\Debug\sd_mem_zed_detect.elf)
```

Figure 13: Target setup in run configuration

- Now Click on Run
- After the Programming is complete , the FPGA is ready for testing

3 Test Procedure

iWave SD Slave IP is tested with Three different hosts ,i.MX Dev Kit , Windows Laptop and Ubuntu Laptop using SD Slot to show the functionality of the IP.

3.1 Tera Term Setup

- Power the i.MX board 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.

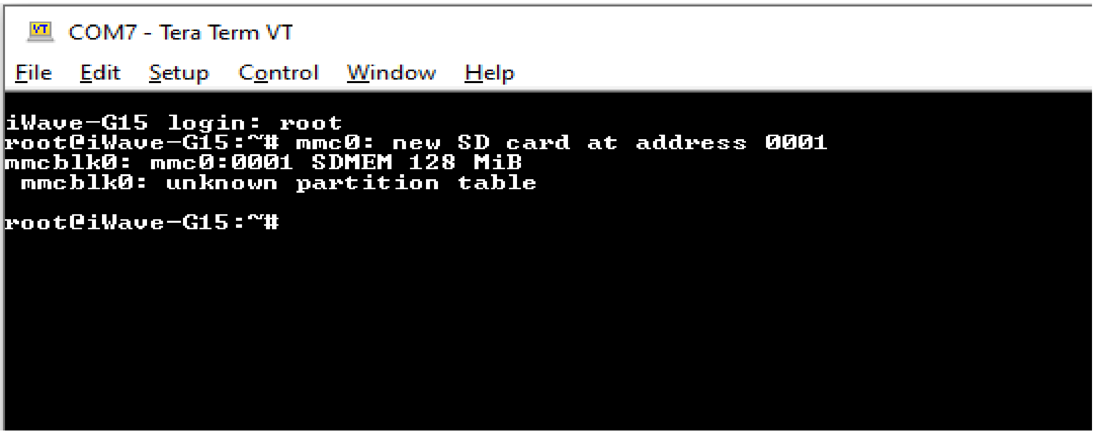
3.2 Testing in i.MX Host

After i.MX Board is completely boot ,Press Enter and Type “root” to login as root user. Then, follow the below procedure to test the SD slave IP.

i.MX Board is used to verify the SD Host IP in ARM based Host.

3.2.1 Card Insertion Test on i.MX Host:

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



```
iWave-G15 login: root
root@iWave-G15:~# mmc0: new SD card at address 0001
mmcblk0: mmc0:0001 SDMEM 128 MiB
 mmcblk0: unknown partition table

root@iWave-G15:~#
```

Figure 14: Card Detection prints for Default Speed

This indicates that 128MB card is detected successfully at the default speed.

3.2.2 Create partition and file system:

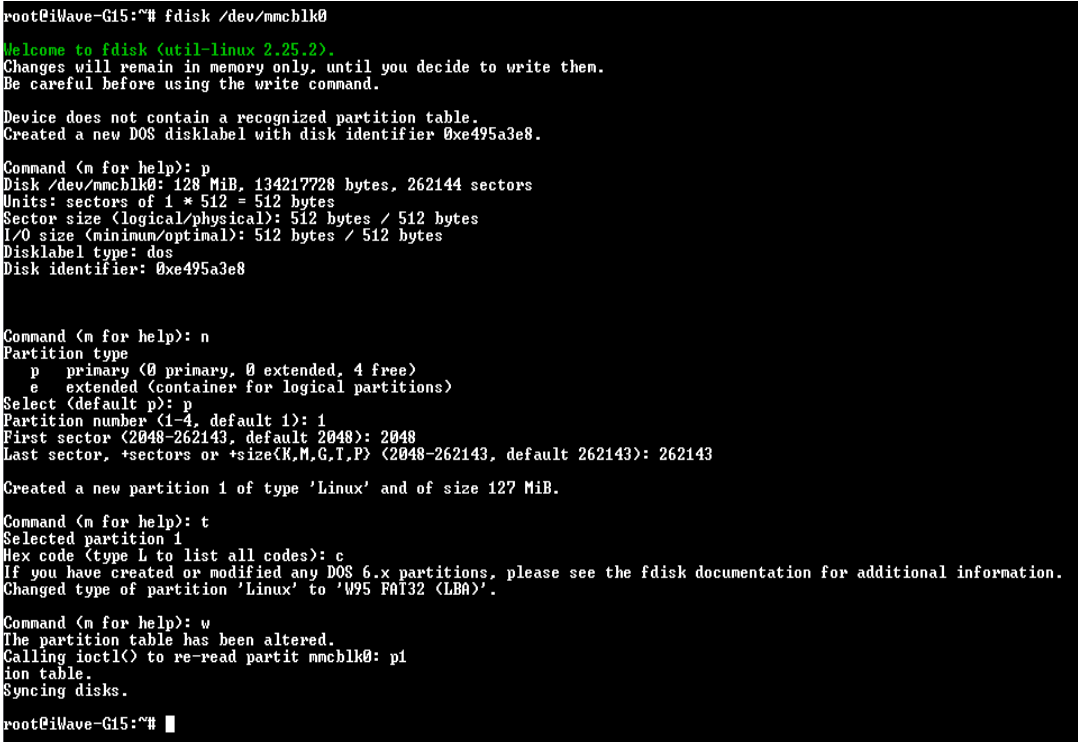
Detected Slave does not contain any partition and file system. To create partition use command

3.2.2.1 fdisk /dev/mmcblk0

Press “***n***” to create new partition.
Press “***p***” to create Primary partition.
Press “Enter” to create 1 partition.
Press “Enter” to create First sector of default size.
Press “Enter” to Create Last Sector of Default Size
After creation of new partition press “***t***” to select file system.
Press ***“l”*** to list all available file system.
Enter “***c***” to select FAT32 filesystem.

After selecting file system press “**w**” to save and exit.

The terminal print is as shown in below figure.



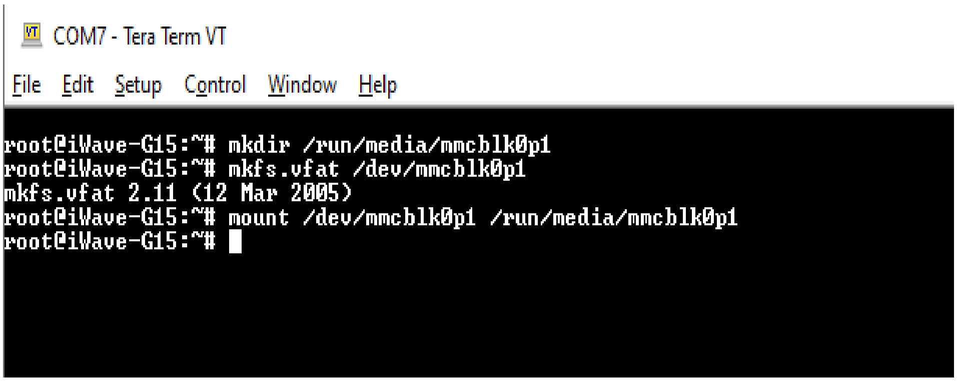
```
root@iWave-G15:~# fdisk /dev/mmcblk0

Welcome to fdisk (util-linux 2.25.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0xe495a3e8.

Command (m for help): p
Disk /dev/mmcblk0: 128 MiB, 134217728 bytes, 262144 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xe495a3e8



Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-262143, default 2048): 2048
Last sector, +sectors or +size{K,M,G,T,P} (2048-262143, default 262143): 262143

Created a new partition 1 of type 'Linux' and of size 127 MiB.

Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): c
If you have created or modified any DOS 6.x partitions, please see the fdisk documentation for additional information.
Changed type of partition 'Linux' to 'W95 FAT32 (LBA)'.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partit mmcblk0: p1
ion table.
Syncing disks.

root@iWave-G15:~# █
```

Figure 15: Creation of partition and selecting file system

After Creating the partition table, mount the partition of SD Slave using the below commands.

3.2.2.2 *mkdir /run/media/mmcblk0p1mkfs.vfat /dev/mmcblk0p1 mount /dev/mmcblk0p1 /run/media/mmcblk0p1*

The terminal print is shown below.



```
root@iWave-G15:~# mkdir /run/media/mmcblk0p1
root@iWave-G15:~# mkfs.vfat /dev/mmcblk0p1
mkfs.vfat 2.11 (12 Mar 2005)
root@iWave-G15:~# mount /dev/mmcblk0p1 /run/media/mmcblk0p1
root@iWave-G15:~# █
```

Figure 16: Mounting the partition

3.2.3 Testing using CP commands in i.MX Host:

After creation of file system mount the SD Slave and then read write any file using cp commands.

"*cp [options] < source file> < destination file*>"

- Where source is root and destination are SD for **write operation**.
- Where source is SD and destination is root for **read operation.**

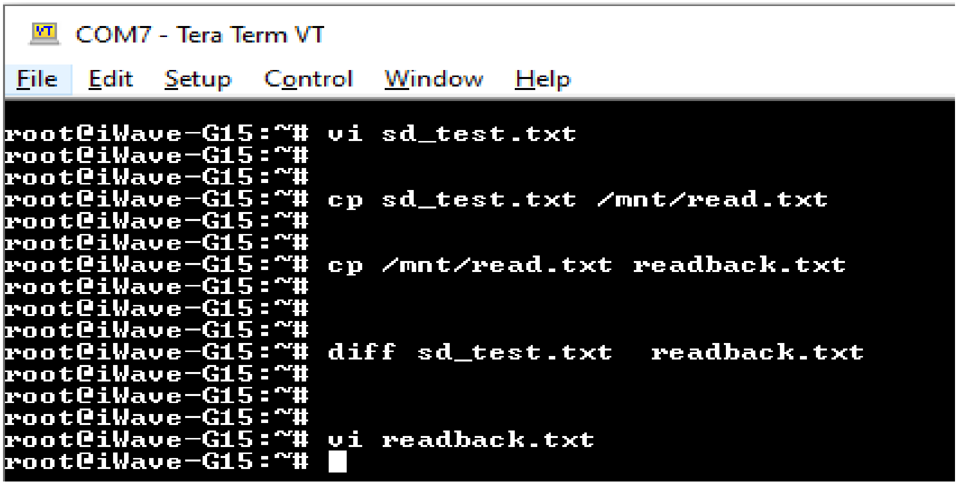
For example: ***"sd_test.txt /mnt/read.txt"*** this will copy the data in the sd_test.txt to into mounted SD Slave to file read.txt

"vi < filename.txt >" command opens the file to edit.

Press ***INSERT*** to edit the file.

After editing the file press ESC and "***:wq***" to save the file.

Following figure shows the prints of copying file using cp commands

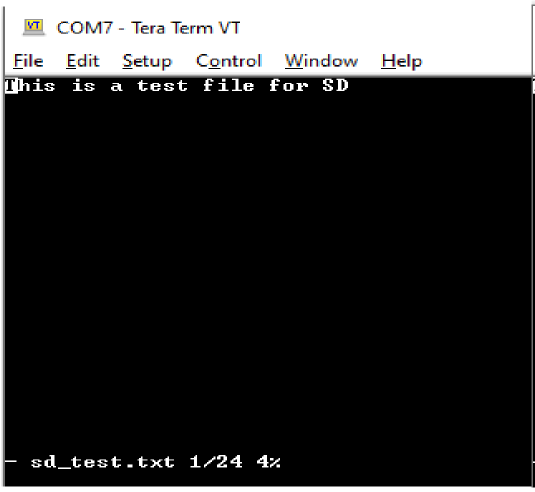


```
root@iWave-G15:~# vi sd_test.txt
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~# cp sd_test.txt /mnt/read.txt
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~# cp /mnt/read.txt readback.txt
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~# diff sd_test.txt  readback.txt
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~#
root@iWave-G15:~# vi readback.txt
root@iWave-G15:~#
```

Figure 17: File Transfer using CP Commands

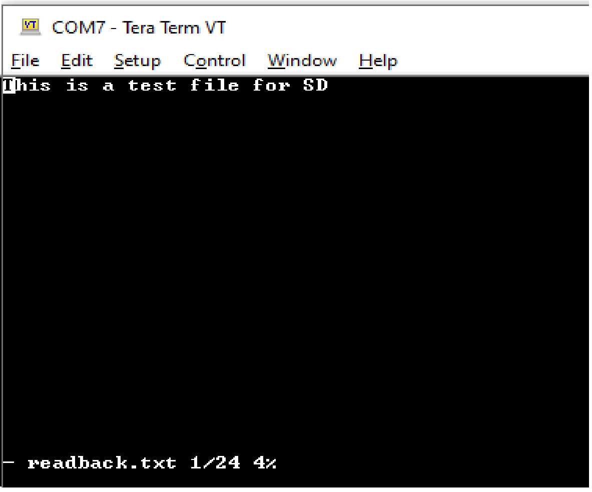
In the above prints **sd_test.txt** file is created, some data are put into it then that file is copied into SD slave as **read.txt** file. This is writing data to SD slave. Then **read.txt** file is copied to root directory as **readback.txt**. This is reading data from SD slave.

To verify the transfer is correct we can check the difference between the file using ***diff*** command as shown in figure. Since the files are transferred properly there is no difference between the files. The below picture shows both source and copied file.



```
This is a test file for SD



- sd_test.txt 1/24 4%
```

Figure 18: Source File



```
This is a test file for SD



- readback.txt 1/24 4%
```

Figure 19: Destination File

3.3 Testing in Windows Host:

Insert the SD Slave extension cable to Windows Laptop SD card Slot and it should be detected as shown in below figure.

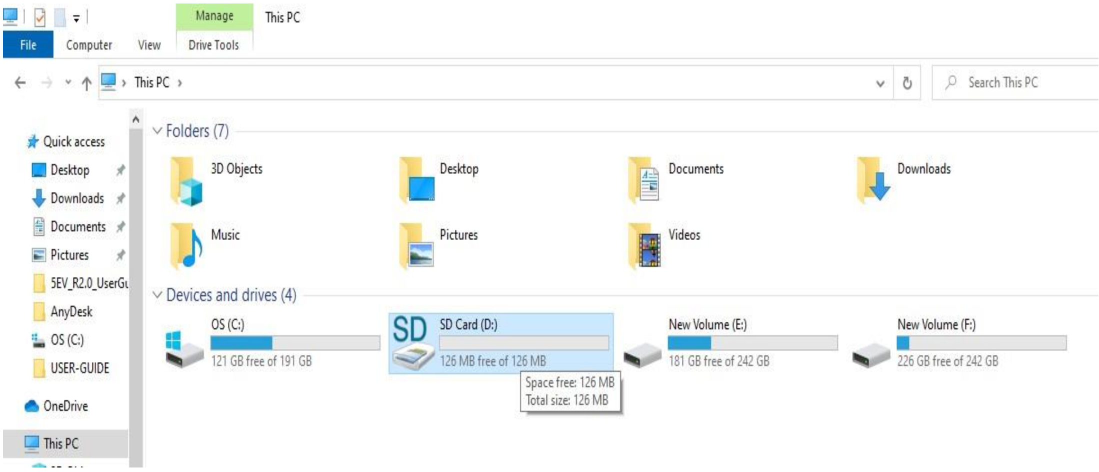


Figure 20: SD Card Detection in Windows PC

Now copy a file from the windows Laptop to the SD Slave , as shown in below image.

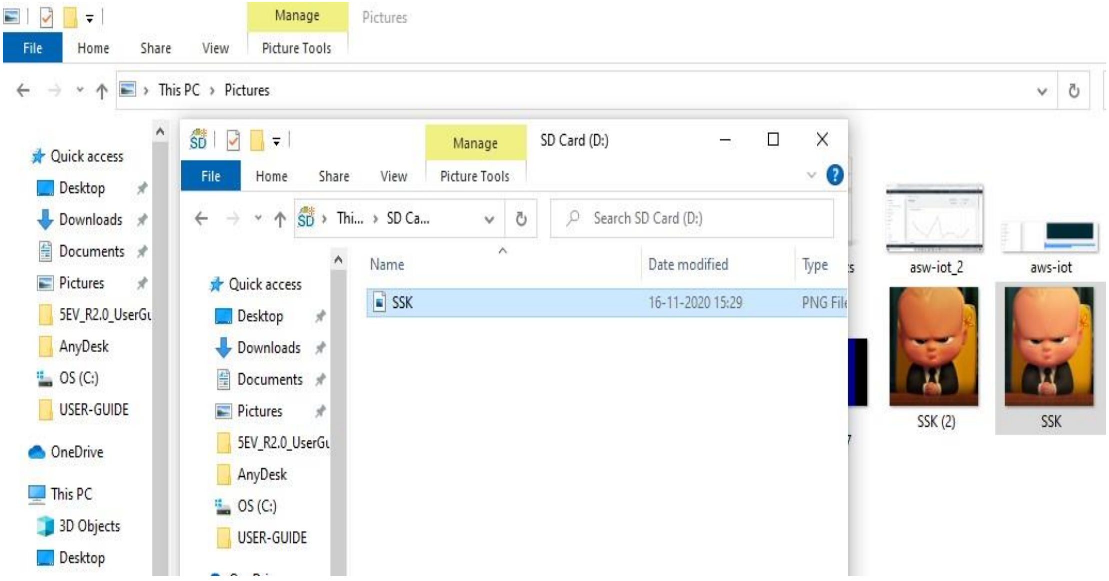


Figure 21: Copying the file from Windows PC to SD Card

Open the file to verify the content .

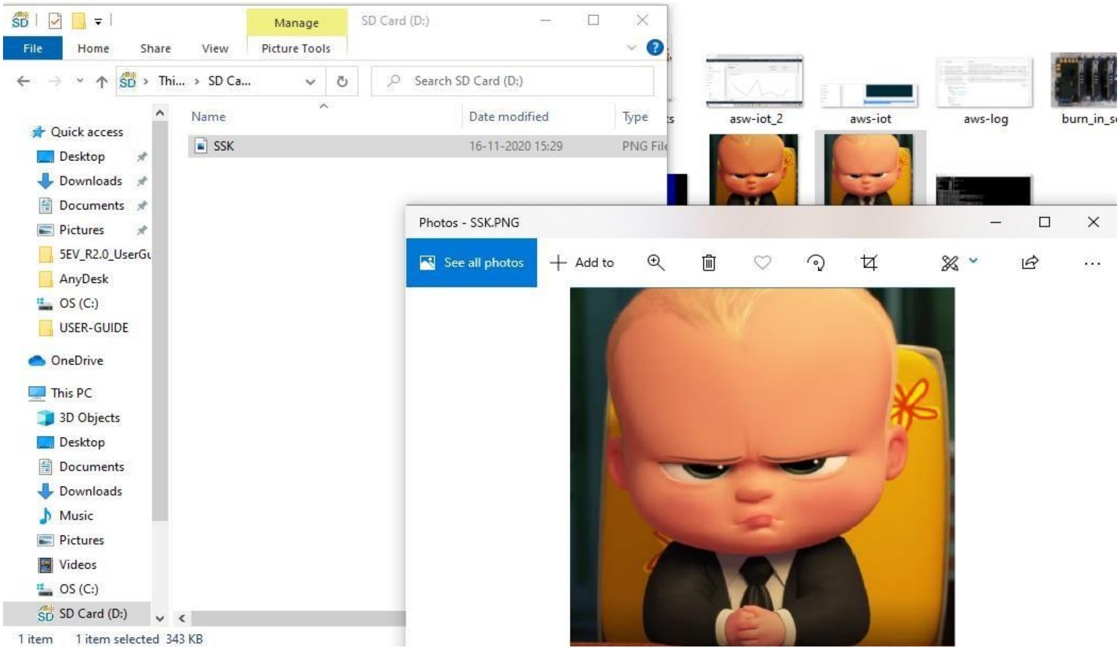


Figure 22: Open the Copied file from the SD Card

3.4 Testing in Linux Host:

Insert the SD extension cable to Linux Laptop SD card Slot and it should be detected as shown in below image.

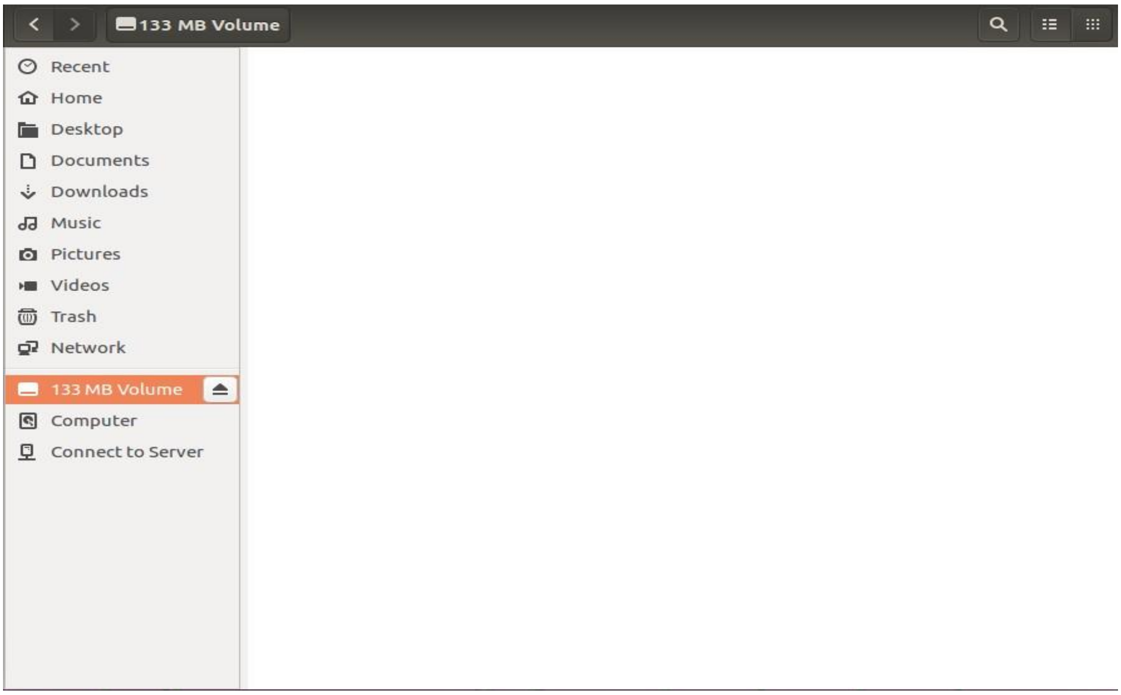


Figure 23: SD Slave detected in Linux Laptop

Now copy any file from the Linux Laptop to the SD slave, as shown in below image.

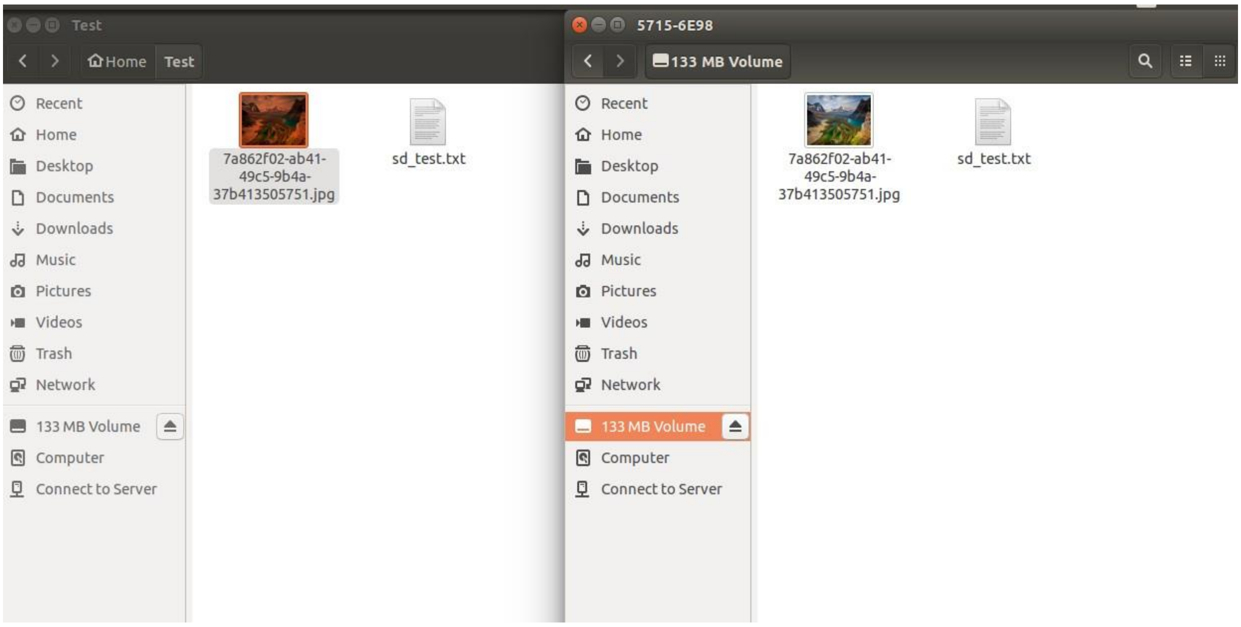


Figure 24: Transfer the files from Linux Laptop to SD Card.

Open the file to verify the content.

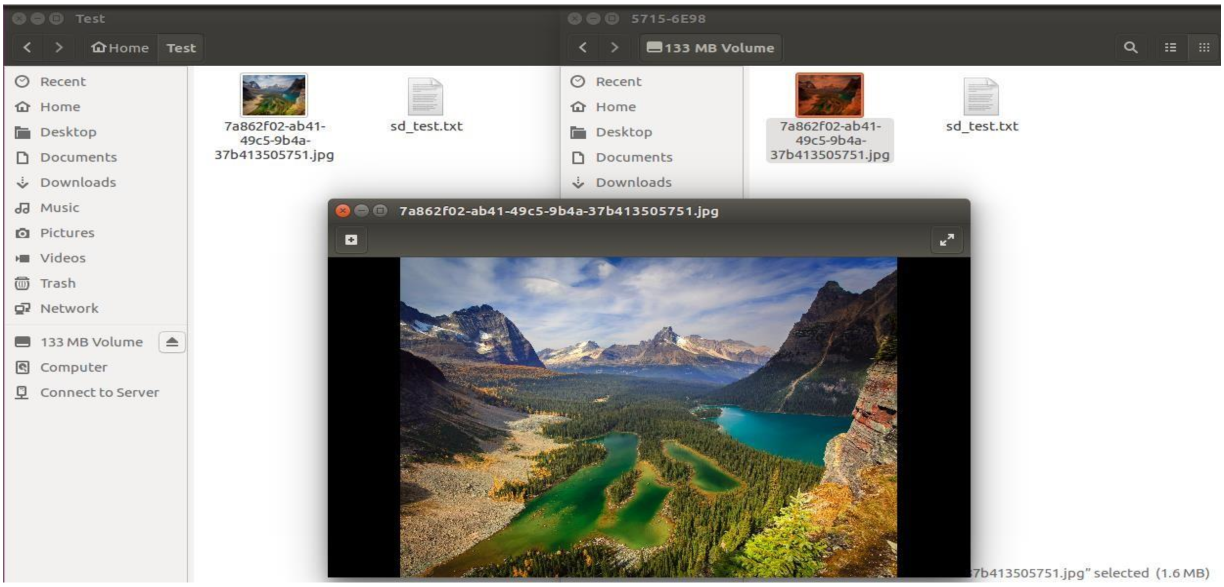


Figure 25: Open the copied file