SD Memory Slave Controller IP Integration Manual

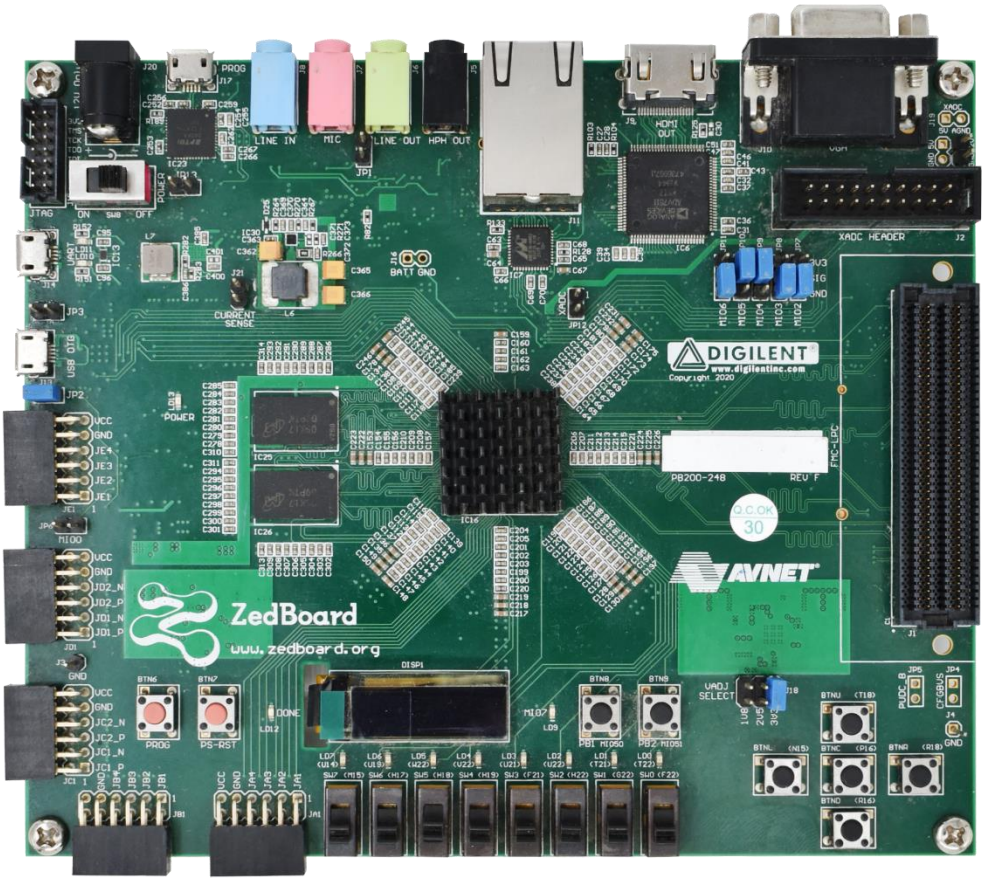

TABLE OF CONTENTS

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.

1.2 Reference Document

- SD Physical layer specification version 3.0.
- iW-EMFDA-DS-01-R1.0-REL1.1

1.3 Overview

SD Memory Slave Controller interfaces the SD host and Processor through AXI4 bus. So, this IP forms a bridge between the SD host and Processor unit, enabling the data transfers between each other. This IP will send the response to the SD host depending on the command issued and also communicates with the Processor through a register interface for control and status.

1.4 Acronyms and Abbreviations

Table 1: Acronyms & Abbreviations

Term	Meaning
FPGA	Field Programmable Gate Array
SD	Secure Digital
AXI	**A**dvanced E**x**tensible **I**nterface
LUT	Look Up Table
IO	Input and Output
FF	Flip Flop

2 IP Configuration and Instantiation

2.1 IP Configuration

The SD Memory Slave IP is configured before synthesis. The configuration of the SD memory Slave controller as either high-Capacity card or standard capacity card. The card status register CSR, SD configuration register SCR, operational condition register OCR and SD status register SSR are all configured pre synthesis. The parameters are preconfigured and synthesized.

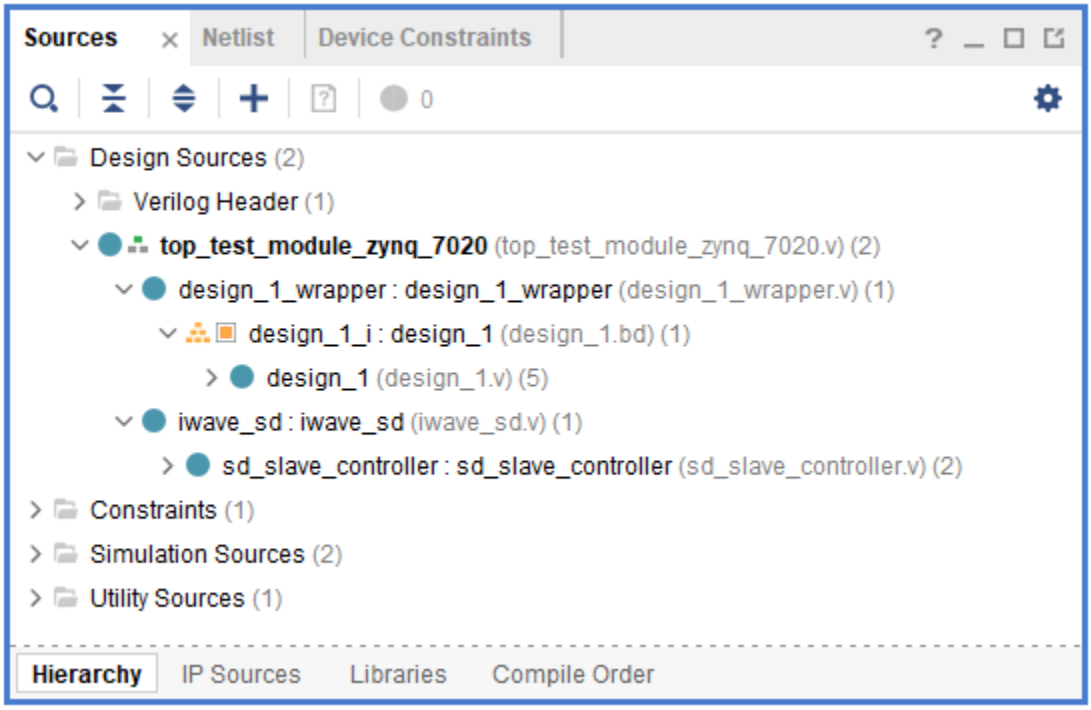

Figure 1: Design sources where SD slave controller IP is instantiated

There are few parameters which can be modified by the user. Parameters like card identification register CID and parameters which decides the memory density of the card are controlled directly from the iwave_sd.v module as shown in the figure 2. The figure 3 shows how to calculate the maximum memory size using the different parameters.

```
MAX_ADDR & MAX_BLK_ADDR parameter calculation will differ for HIGH Capacity
card and Standard capacity card

For High Capacity card, MAX_ADDR = MAX_CAP/512 & MAX_BLK_ADDR = MAX_ADDR
                       MAX_CAP = (C_SIZE_HIGH(in decimal)+1)*512KB
                       For example, with C_SIZE_HIGH = 3FFFh or 16383d
                       MAX_CAP = (16383+1)*512*1024 = 8589934592 = 8GB
                       MAX_ADDR = 8589934592/512 = 16777216d or 1000000h
                       MAX_BLK_ADDR = MAX_ADDR = 16777216d or 1000000h

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 = 3 & 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: Calculations of SD Parameters

```
assign MAX_BLK_ADDR = 32'h40000;            // 128 MB
```

Figure 3: Parameter Controlled from sd_slave_controller.v

```
assign MID          = 8'h0               ;// Manufacturer 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'hfff;            //4095
assign C_SIZE_MULT  = 3'h4;               // for 128M
assign READ_BL_LEN  = 4'h9;               // Corresponds to 512 Bytes
assign WRITE_BL_LEN = 4'h9;               // Corresponds to 512 Bytes
assign MAX_ADDR       = 32'h8000000;      // for 128 MB
```

Figure 4: Parameter Controlled from sd_memory_slave.v

MAX_BLK_ADDR , MAX_ADDR , C_SIZE are Parameters to be changed to vary the s capacity of SD_slave. Parameter is passed from sd_slave_controller.v and sd_memory_slave.v module. Above value is calculated for 128MB Standard Card Size. Refer Figure 3 for calculations.

2.2 Steps to Instantiate SD Memory Slave IP

1) Instantiate the iWave SD Memory Slave Controller IP in module (iwave_sd.v) in custom designs as shown below,

```
sd_slave_controller sd_slave_controller(
  .sd_clk_i    (sd_clk_i      ),
  .sd_data_io  (sd_data_io    ),
  .sd_cmd_io   (sd_cmd_io     ),
  .cd_disable_o(cd_disable_o  ),
  .rst_n_i     (rst_n_i       ),
  .user_clk_i  (user_clk_i    ),
  .user_addr_i (user_addr_i   ),
  .user_data_i (user_data_i   ),
  .user_data_o (user_data_o   ),
  .user_wr_i   (user_wr_i     ),
  .user_rd_i   (user_rd_i     ),
  .user_rvld_o (user_rvld_o   ),
  .user_wdon_o (user_wdon_o   ),
  .user_int_o  (user_int_o    )
);
```

Figure 5: SD Memory Slave IP Instantiation

The SD Interface signals here are to be made external from top module of custom design. Signals with initial SD are SD signals connected to host via SD extension. Signals with initial user are the signals for user interface. SD parameters are the signals given to salve controller from top module.

2) The Constraint file (.xdc) provided in the design can be changed by user depending on requirements.
3) Give the required clock, Pin/IO constraints for SD Interface in the .xdc file and compile the custom design with SD Memory Slave Controller IP.

```
set_property PACKAGE_PIN AA9 [get_ports sd_clk_i]
set_property PACKAGE_PIN AA11 [get_ports sd_cmd_io]
set_property PACKAGE_PIN Y11 [get_ports {sd_data_io[3]}]
set_property PACKAGE_PIN AB10 [get_ports {sd_data_io[2]}]
set_property PACKAGE_PIN AB11 [get_ports {sd_data_io[1]}]
set_property PACKAGE_PIN Y10 [get_ports {sd_data_io[0]}]

set_property IOB TRUE [get_ports sd_cmd_io]
set_property IOB TRUE [get_ports sd_data_io*]

set_property IOSTANDARD LVCMOS33 [get_ports sd_clk_i]
set_property IOSTANDARD LVCMOS33 [get_ports sd_cmd_io]

set_property IOSTANDARD LVCMOS33 [get_ports {sd_data_io[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sd_data_io[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sd_data_io[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sd_data_io[0]}]
```

Figure 5: SD Interface Constraints in .xdc file

Note: The Constraints given in .xdc file are for Zed Board .

3 Implementation Details

3.1 Clock Domain

SD Memory Slave Controller IP uses two clock domains SD clock (sd_clk_i) and user clock (pl_clk0). The SD clock will be obtained from the SD Host interface. The user clock is supplied by the processor.

3.2 Clock constraints

The following figure shows the clock constraints used in this design.
These constraints need to be added in the .xdc file along with the SD Interface pin constraints.

```
create_clock -period 20.000 -name sd_clk_i -waveform {0.000 10.000} [get_ports sd_clk_i]
```

Figure 6: Clock Constraints for High Speed

```
create_clock -period 40.000 -name sd_clk_i -waveform {0.000 20.000} [get_ports sd_clk_i]
```

Figure 7: Clock Constraints for Default Speed

SD Clock (sd_clk_i) is input from the Host and is constrained as shown in the above figure's. Its frequency is 50MHz for high speed and 25 MHz for default speed.

Include below signal drive strength constraints in the .xdc file as shown in the figure below.

```
# DRIVE STRENGTH
set_property DRIVE 16 [get_ports {sd_data_io[3]}]
set_property DRIVE 16 [get_ports {sd_data_io[2]}]
set_property DRIVE 16 [get_ports {sd_data_io[1]}]
set_property DRIVE 16 [get_ports {sd_data_io[0]}]
set_property DRIVE 16 [get_ports sd_cmd_io]

# DRIVE PULLUP
set_property PULLUP true [get_ports sd_cmd_io]
set_property PULLUP true [get_ports {sd_data_io[0]}]
set_property PULLUP true [get_ports {sd_data_io[1]}]
set_property PULLUP true [get_ports {sd_data_io[2]}]
set_property PULLUP true [get_ports {sd_data_io[3]}]
```

Figure 8: Drive Strength Constraints

3.3 Resource Utilization

The table below shows the resource utilization summary for Zed Board xc7z020clg484-1 device with SD 2.0 Memory Slave Controller IP and SD 3.0 Memory Slave Controller IP.

Table 2: Resource Utilization of SD 2.0 Memory Slave Controller IP

Resource	Utilization	Available	Utilization %
LUT	2149	53200	4.04
LUTRAM	60	17400	0.34
FF	2244	106400	2.11
IO	6	200	3.00
BUFG	2	32	6.25

Table 3: Resource Utilization of SD 3.0 Memory Slave Controller IP

Resource	Utilization	Available	Utilization %
LUT	2370	53200	4.45
LUTRAM	60	17400	0.34
FF	2583	106400	2.43
IO	6	200	3.00
BUFG	2	32	6.25

4 Simulation Details

Details of Simulation of SD 3.0 Memory Slave Controller IP is given in this section with the steps.

4.1 Simulation Setup

Simulation is done using ModelSim Software.

1) Provide the Path to the Simulation files in the transcript window , give command do run.do to start the simulation.

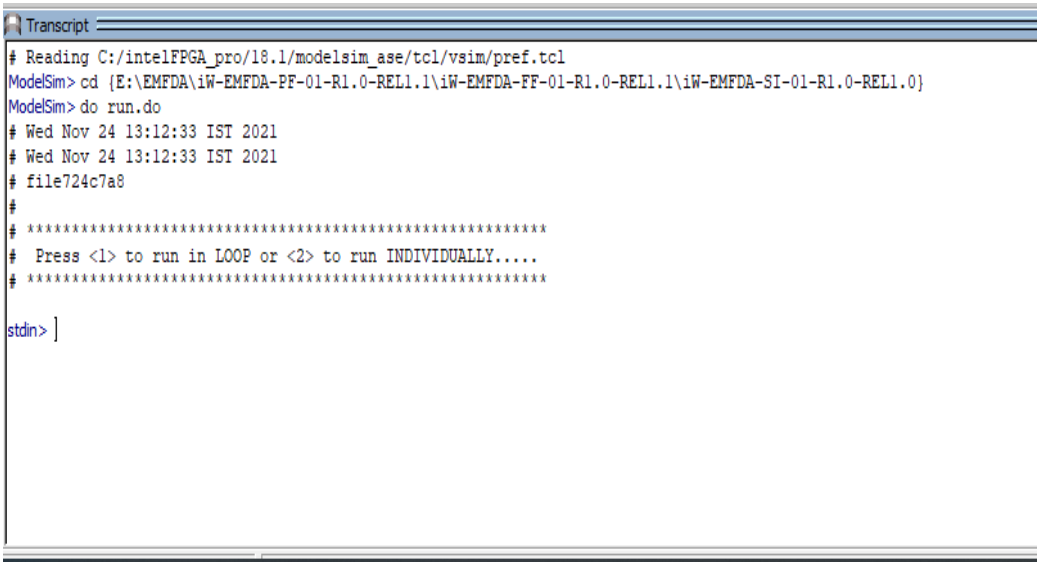

Figure 9: Starting the Simulation

There are 8 testcases , which can be run individually by entering " 2 " or in loop by entering " 1 " in the transcript window. On Entering " 2 " , all the individual test cases are displayed on the window.

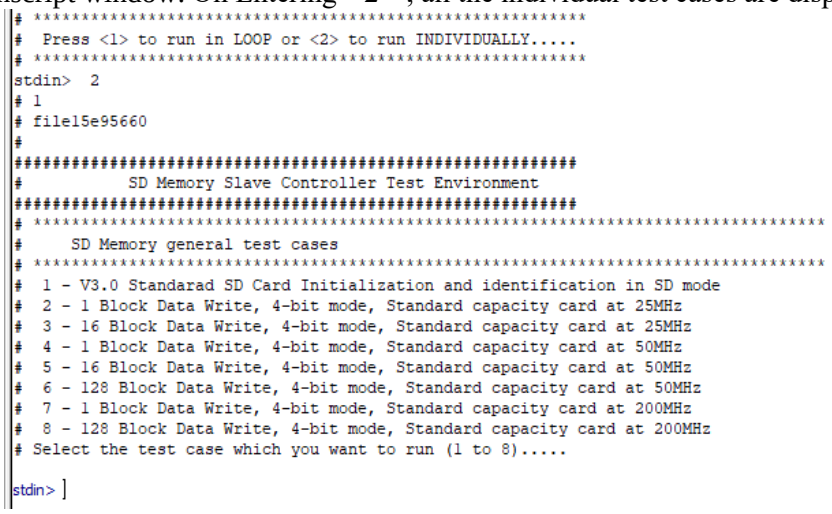

Figure 10: General Test Cases

On Entering valid Test Case , the simulation will start and its completion is indicated in the Transcript window.

```
# CMD Sent is = 12 ####
#
#  ... Start bit of response received
#
#  <<<SD Response received = 0c00004a00b0
#
#  *****SD Index check PASSED *****
#
#  ****SD CRC check PASSED****
#
#  ### TEST PASSED ###
#
#   TEST CASE 8 COMPLETED SUCCESSFULLY
# ********************************************************************************
#
#  All the TEST CASES ARE COMPLETED
#
#  Please open TC_pass_fail.txt to view the results
# ********************************************************************************

VSIM 3>
```

Figure 11: Simulation Results

Once the simulation finishes, check the simulation results by opening the TC_pass_fail.txt file

5 Vitis Changes

Changes are done in " helloworld.c " file variable to support file transfer larger than 32MB.

```
u32 counter=0;
u32 status,sing_mul,addr,flag,sd_status;
u32 src;
u32 old_val=0;
u32 new_val=0;
u32 addr1=0;
u32 err_sdr=0;
u32 err_sdw=0;
u32 a[33554432];// Size of memory/4 = 128 MB /4 = 32 MB = 32*1024*1024
u32 Status;
```

Figure 12: Variable Update

The size of array " a " is increased to " 33554432 ".