SD Memory Slave Controller IP Integration Manual

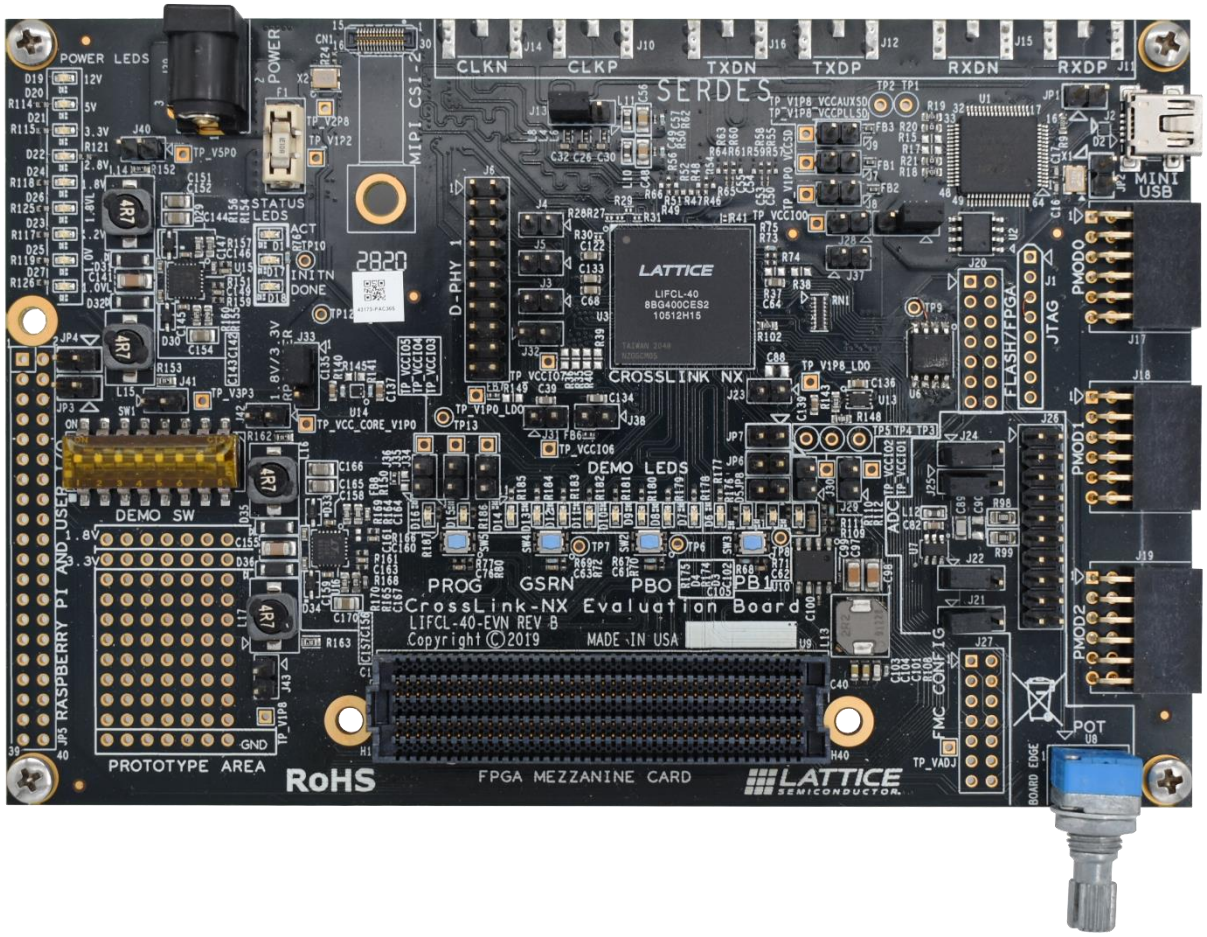

Table of Content

1 INTRODUCTION 5

1.1 PURPOSE 5
1.2 REFERENCE DOCUMENT 5
1.3 OVERVIEW 5
1.4 ACRONYMS AND ABBREVIATIONS 5

2 IP CONFIGURATION AND INSTANTIATION 6

2.1 IP CONFIGURATION 6
2.2 STEPS TO INSTANTIATE SD MEMORY SLAVE IP 6

3 IMPLEMENTATION DETAILS 11

3.1 CLOCK DOMAIN 11
3.2 CONSTRAINTS 11
3.3 FALSE PATH 11

4 RESOURCE UTILIZATION 12

List Of Figures

Figure 1: Design sources where SD slave IP is instantiated 6
Figure 2: ebi_if_top instance part 1 7
Figure 3: ebi_if_top instance part 2 8
Figure 4: sd_mem_ip instance part 1 8
Figure 5: sd_mem_ip instance part 2 9
Figure 6: I/O pin constraints 10
Figure 7: I/O Volatage constraints 10
Figure 8: Clock constraints 11
Figure 9: False path for sd_clk user_clk and native_clk 11

List Of Tables

Table 1: Acronyms & Abbreviations 5
Table 2: Resource Utilization 12

1 Introduction

1.1 Purpose

The purpose of this document is to describe SD Slave Controller IP Integration details on the Lattice Crosslink NX dev 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 IP is instantiated in the design as shown in the figure below.

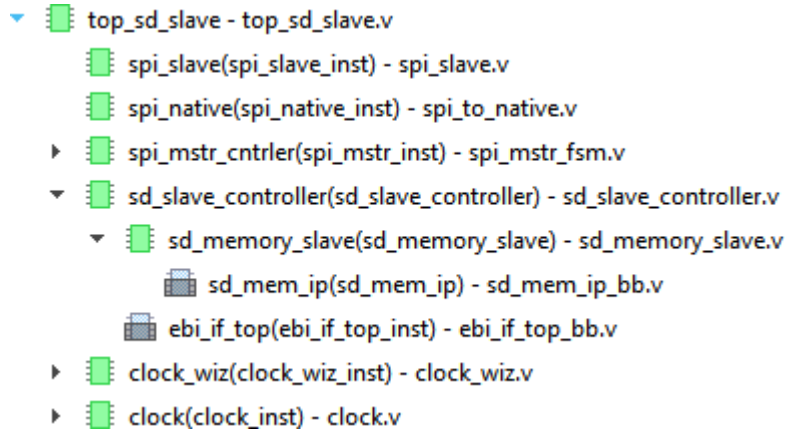

Figure 1: Design sources where SD slave IP is instantiated

Module **top_sd_slave** is the top module of the project which integrates the top module of SD Slave IP with the SPI interface. The Sd_slave controller module has two modules netlisted in it. One is the core SD slave IP i.e., the sd_mem_ip module, and the other is the ebi_if_top module as shown in the figure. To add netlist to the design add sd_mem_ip.vm file and sd_mem_ip_bb.v (stub) file and ebi_if_top.vm file with ebi_if_top_bb.v file to the project and make sure the hierarchy is as shown in the above figure.

2.2 Steps to Instantiate SD Memory Slave IP

1) Instantiate the module ebi_if_top into the project in the sd_slave controller module,

```
ebi_if_top(
 // Clock & Reset
 .native_clk_i            ( native_clk_i         ),
 .sys_clk_i               ( user_clk_i           ),
 .sys_rst_n_i             ( rst_n_i              ),
 .sd_soft_reset_n_i       ( sd_soft_reset_n      ),
 .wrfifo_sd_flush_n_i     ( wrfifo_flush_n       ),
 .rdfifo_sd_flush_n_i     ( rdfifo_flush_n       ),
 .sd_clk_i                ( sd_clk_i             ),
 .global_reset_sd_n_o     ( global_reset_n       ),
 // SD parameter
 .MAX_BLK_ADDR            ( MAX_BLK_ADDR         ),
 // SD Memory Slave Controller
 .xfer_req_i              ( xfer_req             ),
 .xfer_cmd_i              ( xfer_cmd             ),
 .cmd_valid_i             ( cmd_valid            ),
 .command_index_i         ( command_index        ),
 .cmd_arg_i               ( cmd_arg              ),
 .xfer_block_len_i        ( xfer_block_len       ),
 .block_mode_i            ( block_mode           ),
 .stop_cmd_rcvd_i         ( stop_cmd_rcvd        ),
 .data_read_i             ( data_read            ),
 .data_addr_i             ( data_addr            ),
 .wrbuf_we_i              ( wrbuf_we             ),
 .rdbuf_re_i              ( rdbuf_re             ),
 .wr_blk_done_i           ( wr_blk_done          ),
 .wr_blk_crc_err_i        ( wr_blk_crc_err       ),
 .wrbuf_data_i            ( wrbuf_data           ),
 .wrbuf_empty_o           ( wrbuf_empty          ),
 .rdbuf_empty_o           ( rdbuf_empty          ),
 .wr_data_cnt_o           ( wr_data_cnt          ),
 .rd_data_cnt_o           ( rd_data_cnt          ),
 .rdbuf_data_o            ( rdbuf_data           ),
 .write_complete_o        ( write_complete       ),
 .erase_complete_o        ( erase_complete       ),
 .usr_if_error_o          ( usr_if_error         ),
 .no_of_written_block_o   ( no_of_written_block  ),
 .rdy_status_bit_o        ( rdy_status_bit       ),
 // External Bus interface
 .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_intr_o             ( user_int_o           ),
 // Master SPI interface
```

Figure 2: ebi_if_top instance part 1

```
  // Master SPI interface
  .slv_spi_clk_i          ( slv_spi_clk_i       ),
  .cs_n_i                 ( cs_n_i              ),
  .mstr_re_i              ( mstr_re_i           ),
  .trnsfr_dne_i           ( trnsfr_dne_i        ),
  .mstr_actve_i           ( mstr_actve_i        ),
  .data_in_vld_i          ( data_in_vld_i       ),
  .data_in_mstr_i         ( data_in_mstr_i      ),
  .spi_start_o            ( spi_start_o         ),
  .spi_data_out_o         ( spi_data_out_o      ),
  .data_ot_cnt_o          ( data_ot_cnt_o       ),
  .data_vld_o             ( data_vld_o          ),
   // Custom interface
  .cid_reg_o              ( cid_reg             ),
  .rca_reg_o              ( rca_reg             ),
  .csd_reg_o              ( csd_reg             ),
  .scr_reg_o              ( scr_reg             ),
  .ocr_reg_o              ( ocr_reg             ),
  .ssr_reg_o              ( ssr_reg             ),
  .csr_reg_o              ( csr_reg             ),
  .write_event_o          ( write_event_o       ),
  .read_event_o           ( read_event_o        ),
  .stop_event_o           ( stop_event_o        )
);
```

Figure 3: ebi_if_top instance part 2

2) Similarly, instantiate module sd_mem_ip into the project in the sd_memory slave module.

```
sd_mem_ip sd_mem_ip(
  // Clock & Reset
  .sd_clk_i               ( sd_clk_i         ),
  .sys_rst_n_i            ( rst_n_i          ),
  .global_reset_n_i       ( global_reset_n_i ),
  .wrfifo_flush_n_o       ( wrfifo_flush_n_o ),
  .rdfifo_flush_n_o       ( rdfifo_flush_n_o ),
  .sd_soft_reset_n_o      ( sd_soft_reset_n_o),
  .cd_disable_o           ( cd_disable_o     ),
  .sd_data_in_i           ( sd_data_in_i     ),
  .sd_cmd_in_i            ( sd_cmd_in_i      ),
  .sd_data_out_pe_o       ( sd_data_out_pe_o ),
  .sd_data_out_ne_o       ( sd_data_out_ne_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_oen_pe_o       ( sd_data_oen_pe_o ),
  .sd_data_oen_ne_o       ( sd_data_oen_ne_o ),
  // SD parametersMID
  .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_clk_i             ( user_clk_i       ),
  // EBI Interface
```

Figure 4: sd_mem_ip instance part 1

```
  // EBI Interface
  .rdy_staus_bit_i        (rdy_staus_bit_i      ),
  .wrbuf_empty_i          (wrbuf_empty_i        ),
  .rdbuf_empty_i          (rdbuf_empty_i        ),
  .wr_data_cnt_i          (wr_data_cnt_i        ),
  .rd_data_cnt_i          (rd_data_cnt_i        ),
  .rdbuf_data_i           (rdbuf_data_i         ),
  .write_complete_i       (write_complete_i     ),
  .erase_complete_i       (erase_complete_i     ),
  .usr_if_error_i         (usr_if_error_i       ),
  .no_of_written_block_i  (no_of_written_block_i),
  .xfer_req_o             (xfer_req_o           ),
  .xfer_cmd_o             (xfer_cmd_o           ),
  .cmd_valid_o            (cmd_valid_o          ),
  .command_index_o        (command_index_o      ),
  .cmd_arg_o              (cmd_arg_o            ),
  .xfer_block_len_o       (xfer_block_len_o     ),
  .block_mode_o           (block_mode_o         ),
  .stop_cmd_rcvd_o        (stop_cmd_rcvd_o      ),
  .data_read_o            (data_read_o          ),
  .data_addr_o            (data_addr_o          ),
  .wrbuf_we_o             (wrbuf_we_o           ),
  .rdbuf_re_o             (rdbuf_re_o           ),
  .wr_blk_done_o          (wr_blk_done_o        ),
  .wr_blk_crc_err_o       (wr_blk_crc_err_o     ),
  .wrbuf_data_o           (wrbuf_data_o         ),
  .cid_reg_i              (cid_reg_i            ),
  .rca_reg_i              (rca_reg_i            ),
  .csd_reg_i              (csd_reg_i            ),
  .scr_reg_i              (scr_reg_i            ),
  .ocr_reg_i              (ocr_reg_i            ),
  .ssr_reg_i              (ssr_reg_i            ),
  .csr_reg_i              (csr_reg_i            )
);
```

Figure 5: sd_mem_ip instance part 2

After instantiating both the modules check for the hierarchy as shown in figure 1.

3) PDC file is the physical constraint file where all the output pins of the top_sd_slave.v module is declared are shown below

```
############################################################################
#I/O PIN Constraints                                                       #
############################################################################
create_clock -name {sd_clk_i} -period 20.0000 [get_ports sd_clk_i]

ldc_set_location -site {D17} [get_ports cd_disable_o]
ldc_set_location -site {J1}  [get_ports sd_cmd_io]
ldc_set_location -site {K1}  [get_ports sd_clk_i]
ldc_set_location -site {K2}  [get_ports {sd_data_io[0]}]
ldc_set_location -site {K3}  [get_ports {sd_data_io[1]}]
ldc_set_location -site {K4}  [get_ports {sd_data_io[2]}]
ldc_set_location -site {J2}  [get_ports {sd_data_io[3]}]

ldc_set_location -site {P2}  [get_ports stop_event_o   ]
ldc_set_location -site {L2}  [get_ports write_event_o  ]
ldc_set_location -site {R1}  [get_ports read_event_o   ]

#Raspberry PI
ldc_set_location -site {N7} [get_ports sclk_i  ]
ldc_set_location -site {N4} [get_ports mosi_i  ]
ldc_set_location -site {K6} [get_ports miso_o  ]
ldc_set_location -site {P6} [get_ports cs_n_i  ]

# SPI master
ldc_set_location -site {E12} [get_ports spi_clk_o  ]
ldc_set_location -site {D13} [get_ports spi_mosi_o  ]
ldc_set_location -site {D15} [get_ports spi_miso_i  ]
ldc_set_location -site {E13} [get_ports spi_cs_o  ]
```

Figure 6: I/O pin constraints

```
############################################################################
#I/O VOLTAGE Constraints                                                   #
############################################################################
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 PULLMODE=UP} [get_ports rst_n_i]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 PULLMODE=UP} [get_ports sclk_i]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 PULLMODE=UP} [get_ports mosi_i]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 PULLMODE=UP} [get_ports cs_n_i]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 } [get_ports miso_o]

ldc_set_port -iobuf {IO_TYPE=LVCMOS33 } [get_ports sd_clk_i ]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 } [get_ports sd_cmd_io ]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 } [get_ports {sd_data_io[3]}]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 } [get_ports {sd_data_io[2]}]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 } [get_ports {sd_data_io[1]}]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 } [get_ports {sd_data_io[0]}]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 } [get_ports cd_disable_o   ]

ldc_set_port -iobuf {IO_TYPE=LVCMOS33 PULLMODE=UP} [get_ports spi_miso_i ]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 } [get_ports spi_clk_o  ]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 } [get_ports spi_mosi_o  ]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 } [get_ports spi_cs_o  ]


ldc_set_port -iobuf {IO_TYPE=LVCMOS33 } [get_ports stop_event_o   ]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 } [get_ports write_event_o  ]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33 } [get_ports read_event_o   ]
```

Figure 7: I/O Volatage constraints

3 Implementation Details

3.1 Clock Domain

SD Slave Controller IP works at clock speed 25Mhz for default speed configuration or 50 MHz High-speed configurations. User clock input whose frequency is 100MHz and Native clock is of 200MHz frequency.

3.2 Constraints

The following figure shows the clock constraints used in this design. The constraints are given herefor the SD clock whose frequency is 50Mhz for High speed.
It is defined in the "**impl_1.pdc"** file

```
create_clock -name {sd_clk_i} -period 20.0000 [get_ports sd_clk_i]
```

Figure 8: Clock constraints

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

3.3 False path

```
##############################################################################
# False path                                                                 #
##############################################################################

set_false_path -from [get_clocks user_clk ] -to [get_clocks sd_clk_i]
set_false_path -from [get_clocks sd_clk_i] -to [get_clocks user_clk ]
set_false_path -from [get_clocks user_clk ] -to [get_clocks native_clk ]
set_false_path -from [get_clocks native_clk ] -to [get_clocks user_clk ]
set_false_path -from [get_clocks native_clk ] -to [get_clocks sd_clk_i]
set_false_path -from [get_clocks sd_clk_i] -to [get_clocks native_clk ]
```

Figure 9: False path for sd_clk user_clk and native_clk

The above figure shows the flash path constraints added to indicate the clock domain crossingsignals are properly taken care of and the tool can ignore the CDC.

4 Resource Utilization

The table below shows the resource utilization summary for **Lattice Crosslink NZ** LIFCL-40-9BG400C device with SD Memory Slave Customization.

Table 2: Resource Utilization

Resource	Utilization
LUT	5680
Register	6710
EBR	16
IO Buffers	22