NAND Flash Controller Integration Manual

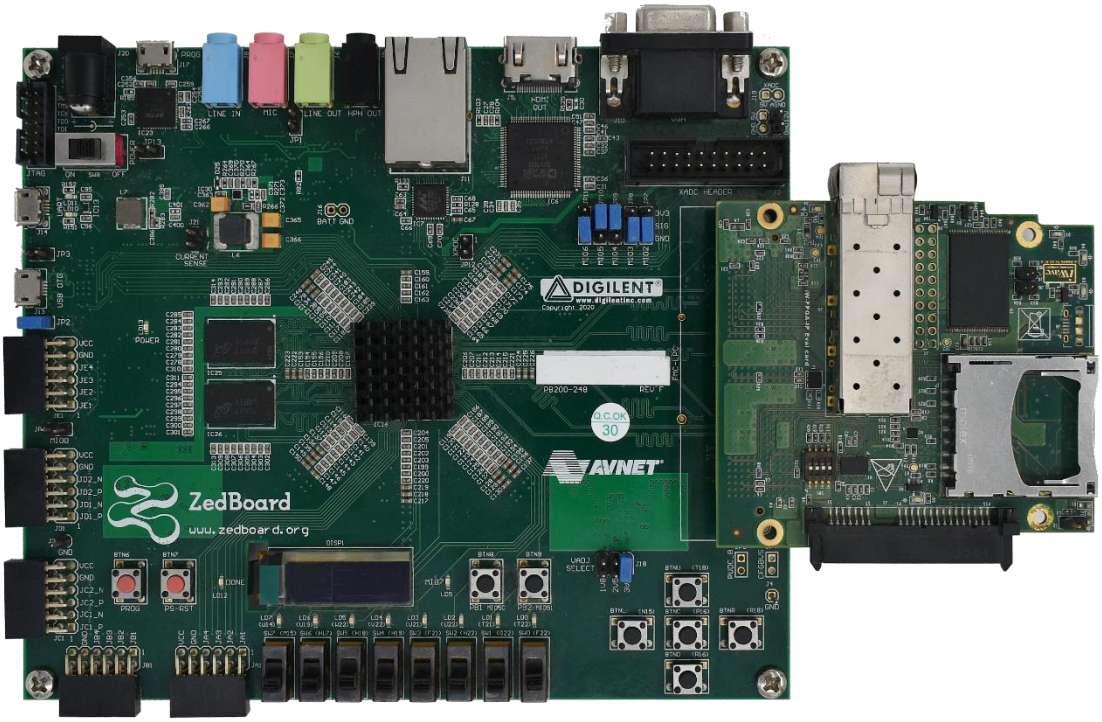

TABLE OF CONTENTS

List of Figures

List of Tables

1 Introduction

1.1 Purpose

The purpose of this document is to describe NAND Flash Controller IP Integration details.

1.2 Scope

This document describes the NAND Flash Controller IP and provides information about IP integration details.

1.3 Reference Document

- NAND Flash Memory Datasheet

  https://datasheetspdf.com/datasheet/MT29F64G08AFAAA.html
- Open NAND Flash Interface Specification
- AMBA® AXI™ and ACE™ Protocol Specification

1.4 Overview

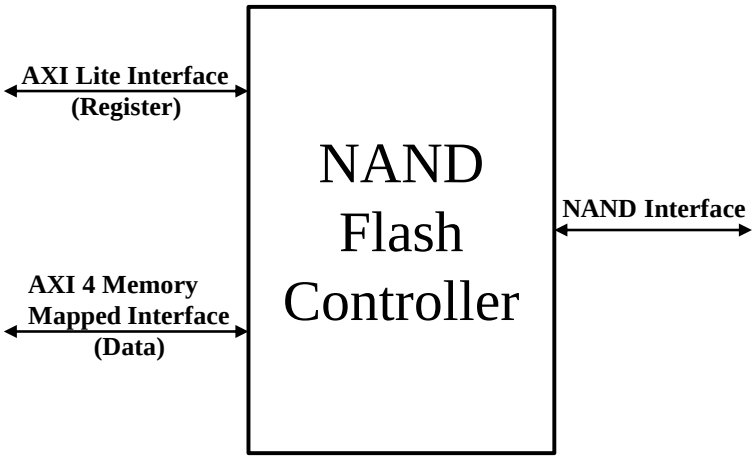

Figure 1: NAND Flash Controller Block Diagram

NAND Flash Controller defines a standard register set for control and status of the NAND device. iWave Supplies NAND Flash Controller as standalone component in addition to Verilog/VHDL modules

1.5 Acronyms and Abbreviations

Table 1: Acronyms & Abbreviations

Term	Meaning
AXI	Advanced eXtensible Interface
FPGA	Field Programmable Gate Array
DDR	Double Data Rate
HDL	Hardware Description Language
IP	Intellectual Property
irq	Interrupt
ns	Nano seconds
PL	Programmable Logic
PS	Processor System
RTL	Register Transfer Logic
SoC	System On Chip
us	Micro Seconds

2 IP Configuration and Instantiation

2.1 Example design

The NAND flash IP example design mainly consists of,

1. **NAND HOST IP:** NAND HOST IP is the design under test and controls the access to the external NAND chip based on the inputs from register set module.
2. **NAND Wrapper:** This module accepts the commands and data over AXI 4 memory mapped interface and translates them to the user interface of NAND Host
3. **NAND Register Set:** This module implements register set required for NAND Device control and status information
4. **Clock Enable Generator:** This module generate clock enable for the NAND Host.
5. **Design Wrapper:** This Wrapper module consist of the design units along with the Zynq 7 Processing System.

Note:

- **Example design is compiled for XC7Z045FFG900-2 device with Vivado 2019.2 version.**
- **Example design currently supports dual die.**

An example design project is created using Vivado 2019.2, at location ..\iW-EMFEY-Release-1.0\iW- EMFEY -Release-1.0\iW- EMFEY -PF-01-R1.0-REL1.0\iW-EMFEY -FF-01-R1.0\iW- EMFEY -SY-01-R1.0-REL1.0\iW_ EMFEY _ED.
The design units that make the Block Design are shown in the figure below:

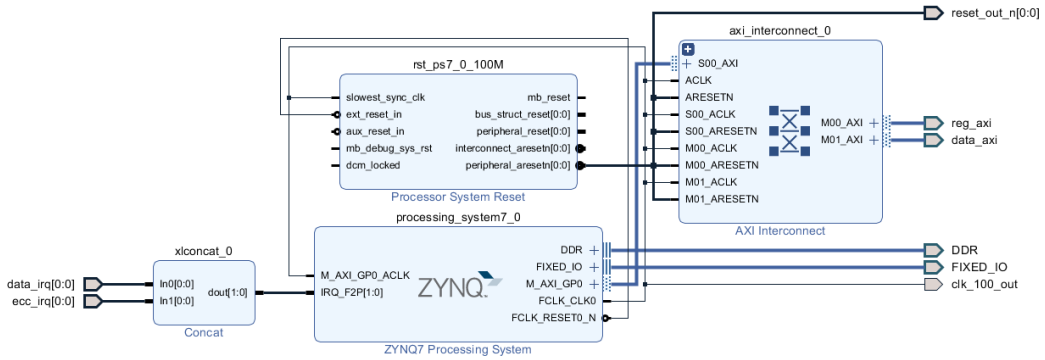

Figure 2: Block Design Modules of the example design project

Note: Change the assigned addresses in Address Editor tab present along with the block design/Diagram as below

1) For the register access: reg_axi port assign 0x6800_000 as address and range to 8M

2)For the data transfer: data_axi port assign 0x5000_0000 as address and range to 256M

data_axi	data_axi	Reg	0x5000_0000	256M	0x5FFF_FFFF
reg_axi	reg_axi	Reg	0x6800_0000	8M	0x687F_FFFF

Figure 3: Assign Specific Addresses & Range to ports

2.2 IP Configuration

The Nand Flash Host Controller IP includes the netlist file i.e. nand_host.dcp in the design as shown in the figure below

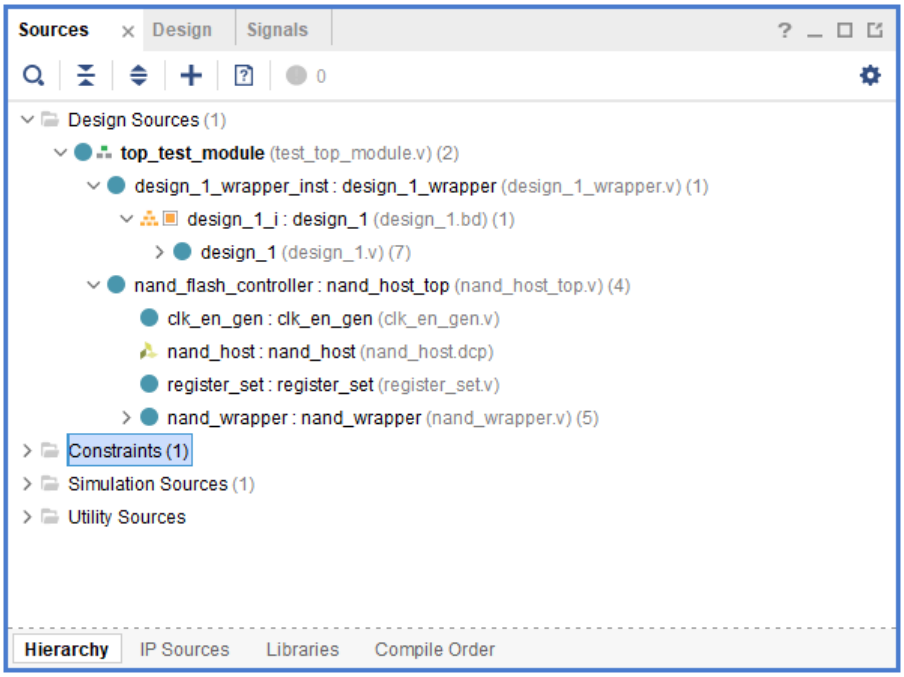

Figure 4: Design sources where nand_host netlist is instantiated

Top Module of the Example design project is the top_test_module that instantiates Nand_flash_controller IP and wrapper file of the block design i.e design_1.
Nand_host(.dcp file),register_set,nand_wrapper and clk_en_gen are instantiated in Nand_flash_controller.

Create a top module as shown in Figure below

```
module top_test_module #(
  // NAND Device Timing Parameters
   parameter TPWR         = 20'd1000  ,
   parameter TRST         = 20'd50000 ,
   parameter TR_ECC       = 20'd7000  ,
   parameter TR           = 20'd7000  ,
   parameter TPROG        = 20'd60000 ,
   parameter TBERS        = 20'd300000,
   parameter TWB          = 20'd100   ,
   parameter TFEAT        = 20'd100   ,
   parameter TWHR         = 20'd20    ,
   parameter TCCS         = 20'd20    ,
   // NAND Device Page Parameters
   parameter DEV_PG_SIZE  = 13'h2000  , // Page Size 8K
   parameter DEV_OOB_SIZE = 09'h1C0     // Spare Size 440 bytes
  )
  (
  //PS DDR Signals
    inout  [14:0]        DDR_addr,
    inout  [2:0]         DDR_ba,
    inout                DDR_cas_n,
    inout                DDR_ck_n,
    inout                DDR_ck_p,
    inout                DDR_cke,
    inout                DDR_cs_n,
    inout  [3:0]         DDR_dm,
    inout  [31:0]        DDR_dq,
    inout  [3:0]         DDR_dqs_n,
    inout  [3:0]         DDR_dqs_p,
    inout                DDR_odt,
    inout                DDR_ras_n,
    inout                DDR_reset_n,
    inout                DDR_we_n,
    inout                FIXED_IO_ddr_vrn,
    inout                FIXED_IO_ddr_vrp,
    inout  [53:0]        FIXED_IO_mio,
    inout                FIXED_IO_ps_clk,
    inout                FIXED_IO_ps_porb,
    inout                FIXED_IO_ps_srstb,
    //NAND Flash Interface
    input                nand_r_b_0_n_i     , // RDY/BSY#
    input                nand_r_b_1_n_i     , // RDY/BSY#
    output               nand_ce_0_n_o      , // Chip Enable
    output               nand_ce_1_n_o      , // Chip Enable
    output               nand_re_n_o        , // Read Enable
    output               nand_we_n_o        , // Write Enable
    output               nand_cle_o         , // Command Latch Enable
    output               nand_ale_o         , // Address Latch Enable
    output               nand_wp_n_o        , // Write Protect
    inout   [7:0]        nand_data_io         //NAND data
  );
```

Figure 5:Top Module for the example design

2.3 Steps to Instantiate NAND HOST IP

1. Instantiate the Nand Host IP in the NAND Flash Controller with the IO signals of Nand Host IP as shown in Figure 6.
2. Instantiate the NAND Flash Controller IP and the generated Block Design Wrapper in the top module of the example design.
3. Make Signal connections between the NAND Flash Controller and the block diagram wrapper in the top_test_module of the example design.

```
// NAND Host Controller
nand_host nand_host (
  // Timing Parameters
  .TPWR                (TPWR            ),
  .TRST                (TRST            ),
  .TR_ECC              (TR_ECC          ),
  .TR                  (TR              ),
  .TPROG               (TPROG           ),
  .TBERS               (TBERS           ),
  .TWB                 (TWB             ),
  .TFEAT               (TFEAT           ),
  .TWHR                (TWHR            ),
  .TCCS                (TCCS            ),
  .DEV_PG_SIZE         (DEV_PG_SIZE     ),
  .DEV_OOB_SIZE        (DEV_OOB_SIZE    ),
  // Clock, Clock Enable and Reset
  .clk_i               (s_aclk          ),
  .rst_n_i             (s_aresetn       ),
  .clk_en_i            (clk_en          ),
  .ce_n_i              (ce_n            ),
  // ECC Signals
  .disable_ecc_i       (disable_ecc     ),
  .ecc_last_i          (ecc_last        ),
  .ecc_error_o         (ecc_fail        ),
  .ecc_read_o          (ecc_read        ),
  .ecc_active_o        (ecc_active      ),
  .ecc_busy_o          (ecc_busy        ),
  .ecc_complete_o      (ecc_done        ),
  // Command & Address Signals
  .cmd_req_i           (command_req     ),
  .wp_n_i              (write_protect_n ),
  .usr_cmd_i           (usr_cmd         ),
  .end_command         (end_command     ),
  .addr_i              (nand_address    ),
  .addr_cyc_i          (no_addr_cycle   ),
  .data_size_i         (data_size       ),
  .cmd_rdy_o           (command_rdy     ),
  .err_o               (err             ),
  // User Data Interface
  .rd_data_vld_o       (read_dvalid     ),
  .rd_data_o           (read_data       ),
  .rd_ready_i          (read_ready      ),
  .wr_data_vld_i       (write_dvalid    ),
  .wr_data_i           (write_data      ),
  .wr_ready_o          (write_ready     ),
  // ECC Value Signals
  .value0_vld_o        (value0_vld      ),
  .value0_corrected_o  (value0_corrected),
  .ecc_value0_o        (ecc_value0      ),
  .value1_vld_o        (value1_vld      ),
  .value1_corrected_o  (value1_corrected),
  .ecc_value1_o        (ecc_value1      ),
  .value2_vld_o        (value2_vld      ),
  .value2_corrected_o  (value2_corrected),
  .ecc_value2_o        (ecc_value2      ),
  .value3_vld_o        (value3_vld      ),
  .value3_corrected_o  (value3_corrected),
  .ecc_value3_o        (ecc_value3      ),
  // NAND Device Interface Signals
  .nand_r_b_n_i        (nand_r_b_n_i    ),
  .nand_ce_n_o         (nand_ce_n       ),
  .nand_re_n_o         (nand_re_n_o     ),
  .nand_we_n_o         (nand_we_n_o     ),
  .nand_cle_o          (nand_cle_o      ),
  .nand_ale_o          (nand_ale_o      ),
  .nand_wp_n_o         (nand_wp_n_o     ),
  .nand_data_en_o      (nand_data_en_o  ),
  .nand_data_i         (nand_data_i     ),
  .nand_data_o         (nand_data_o     ),
  .nand_data_dir_o     (nand_data_dir_o )
);
```

Figure 6: Instantiation of NAND Host IP in NAND Flash Controller module

4. Give the pin constraint for NAND interface using .xdc file as shown in Figure 7.
5. Save the project and Generate the Bit stream file.

3 Implementation Details

3.1 Parameter Settings

NAND IP consists of several parameters. These are to be set before compiling the project. The values of the parameters can be found from the NAND Datasheet. Following are the parameters required by the NAND Host Controller.

1. TPWR : Power-On Recovery Time
2. TRST : Device Resetting Time
3. TR_ECC : Data Transfer Time Flash Array to Data Register, Internal ECC Enabled
4. TR : Data Transfer Time Flash Array to Data Register, Internal ECC Disabled
5. TPROG : PROGRAM PAGE Operation Time
6. TBERS : BLOCK ERASE Operation Time
7. TWB :WE# HIGH to Busy
8. TFEAT :Busy Time for FEATURES command operations
9. TWHR :WE# HIGH to RE# LOW
10. TCCS :Column Address Set Time
11. DEV_PG_SIZE :Page size of the NAND device
12. DEV_OOB_SIZE:OOB size of the NAND device

Most of the parameters are given as the maximum time; these have to be considered as it is. Parameters given with minimum time, additionally consider a buffer time of 50-100ns. Get the values of the timing parameters (ns or us) from the datasheet and using the following example apply all the values to the above parameters. Consider the TRST time, in datasheet this time is defined as 500us (MAX). Based on the clock configured say, it is 100MHz (10ns). Then set the TRST parameter with value = 500us/10ns = 50000d If clock configured is 50MHz(20ns), set the TRST as 500us/20ns = 25000d, Similarly other parameters are to be added before compilation, in the file top_test_module.v.

NOTE: All the above parameters should be with respect to the frequency of 100MHz and values shown is for Micron MT29F64G08AFAAA part.

3.2 Constraints

Following figure shows the example pin and IO standard constraints.

```
set_property PACKAGE_PIN AC24 [get_ports nand_r_b_0_n_i]
set_property PACKAGE_PIN AH22 [get_ports nand_r_b_1_n_i]
set_property PACKAGE_PIN AG24 [get_ports nand_ce_0_n_o]
set_property PACKAGE_PIN AG25 [get_ports nand_ce_1_n_o]
set_property PACKAGE_PIN AD24 [get_ports nand_re_n_o]
set_property PACKAGE_PIN AD21 [get_ports nand_we_n_o]
set_property PACKAGE_PIN AF23 [get_ports nand_cle_o]
set_property PACKAGE_PIN AF24 [get_ports nand_ale_o]
set_property PACKAGE_PIN AE21 [get_ports nand_wp_n_o]
set_property PACKAGE_PIN AG20 [get_ports {nand_data_io[7]}]
set_property PACKAGE_PIN AF20 [get_ports {nand_data_io[6]}]
set_property PACKAGE_PIN AH21 [get_ports {nand_data_io[5]}]
set_property PACKAGE_PIN AG21 [get_ports {nand_data_io[4]}]
set_property PACKAGE_PIN AF22 [get_ports {nand_data_io[3]}]
set_property PACKAGE_PIN AE22 [get_ports {nand_data_io[2]}]
set_property PACKAGE_PIN AE23 [get_ports {nand_data_io[1]}]
set_property PACKAGE_PIN AD23 [get_ports {nand_data_io[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports nand_r_b_0_n_i]
set_property IOSTANDARD LVCMOS33 [get_ports nand_r_b_1_n_i]
set_property IOSTANDARD LVCMOS33 [get_ports nand_ce_0_n_o]
set_property IOSTANDARD LVCMOS33 [get_ports nand_ce_1_n_o]
set_property IOSTANDARD LVCMOS33 [get_ports nand_re_n_o]
set_property IOSTANDARD LVCMOS33 [get_ports nand_we_n_o]
set_property IOSTANDARD LVCMOS33 [get_ports nand_cle_o]
set_property IOSTANDARD LVCMOS33 [get_ports nand_ale_o]
set_property IOSTANDARD LVCMOS33 [get_ports nand_wp_n_o]
set_property IOSTANDARD LVCMOS33 [get_ports {nand_data_io[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {nand_data_io[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {nand_data_io[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {nand_data_io[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {nand_data_io[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {nand_data_io[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {nand_data_io[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {nand_data_io[7]}]
```

Figure 7: Pin Constraints in example design .xdc file

NOTE:
These constraints are in accordance to example design for XC7Z045FFG900-2 FPGA device. Please change the pin constraints for NAND interface for other custom board. Also define the constraints for clock, reset and UART pins accordingly if required for custom board.
GPIO Pins can be used in .xdc file if required as shown in Figure 7.

4 FPGA Implementation

4.1 Resource Utilization

The table below shows the utilization summary from the implementation of NAND Flash Host Controller for Zynq-7000 SoC FPGA device with part number XC7Z045FFG900-2.

Table 2: Device Utilization Summary for XC7Z045FFG900-2

Resource	Utilization	Available
LUT	3096	218600
FF	2235	437200
BRAM	13	545