NCTL Design Spec

Version 1.01

2023/3/9

云海芯科微电子科技有限公司

CONTENTS

FIGURE

TABLE

REVISION HISTORY

Revision	Date	Modified By	Description
1.00	2023/03/02	James	Initial Release
1.01	2023/03/09	James	Add 0x1050~0x108f register description

Revision m.xy, where:

- m the first digit are incremented for major changes of substance, e.g., function changes.
- xy the second two digits are incremented when minor changes have been incorporated into the specification, i.e., enhancements, corrections, updates, etc.

1. INTRODUCTION

1.1. Document Scope

NCTL (NAND Controller) is the Flash controller of high performance PCIe Gen5 SSD. This Design Spec. would describe the design principles of the NCTL, interface signal and registers in detail.

1.2. Reference

- ONFI 5.1 PHY Design Spec
- BMU(Buffer Management) Design Spec
- Firmware AHB-like interface

1.3. Glossary

- PWT – Paradigm Works Training
- DUT – Device Under Test
- CPU – Central Processing Unit
- Packet – A custom datagram used to send and receive data from and to a device
- RO – Read Only
- RW – Read/Write
- W1C – Read/Write 1 to Clear
- RC – Read to Clear

1.4. Design Information

Table 1. Product List

Part Number	Application	Package Type	Package Size (mm)	Flash Die

1.5. Design Feature

- Provide 3072 command queues, and automatically select the appropriate command to execute according to the command characteristics and priority

- According to the LBA and multi stream ID of each 4KB command, the hardware automatically determines whether it needs to be combined into a NAND command for execution
- Hardware handles NAND functions such as Cache read, Cache write, Suspend, Resume, and AIPR/IWL
- According to different NAND vendors, built-in scramble function
- Built-in Bad Block Remapping function, automatically remapping according to Bad Block position
- Each channel has a built-in Micro Processor, which automatically processes the NAND command sequence
- The hardware can flexibly set the timing constraint of each NAND command
- Built-in system clock & NAND clock asynchronous processing interface

2. DESIGN DESCRIPTION

2.1. NCTL Design Overview

NCTL 是 NAND Controller 的縮寫，主要負責處理 command 的 priority，並在 BMU 跟 ONFI PHY 之間傳遞資料. NCTL 大部分的 module 都是 synchronous design. 僅在 BMU interface 經過 asynchronous 處理. NCTL 首先從 AHB-like(register interface)接收 Firmware 寫入的 Read/Write command 以及此 command 相關的資訊，在 parsing command 後，會根據此 command 的 channel, CE 以及 priority，將 command 存入對應的 command FIFO 中. 接著再根據每個 NAND 的 ready/busy 狀態，從對應的 command FIFO 中取出要讀寫的 block/page/frag, 並發送 command 到 Micro processor. Micro processor 是一個自研的簡易處理器，負責從 chan_reg 接收指令，並從 share_sram 將 instruction 讀出，再根據每一個 instruction 的內容，做適當的處理，最後將 CMD/ADDR/DATA 寫入 protocol handler (ptc_hdl.v) module，以產生 NAND command sequence. ptc_hdl 同時也內建 DMA engine，可透過 memory read/write interface，將 data 從 BMU 讀出並寫入 ONFI PHY，或是將 data 從 ONFI PHY 讀出並寫入 BMU.

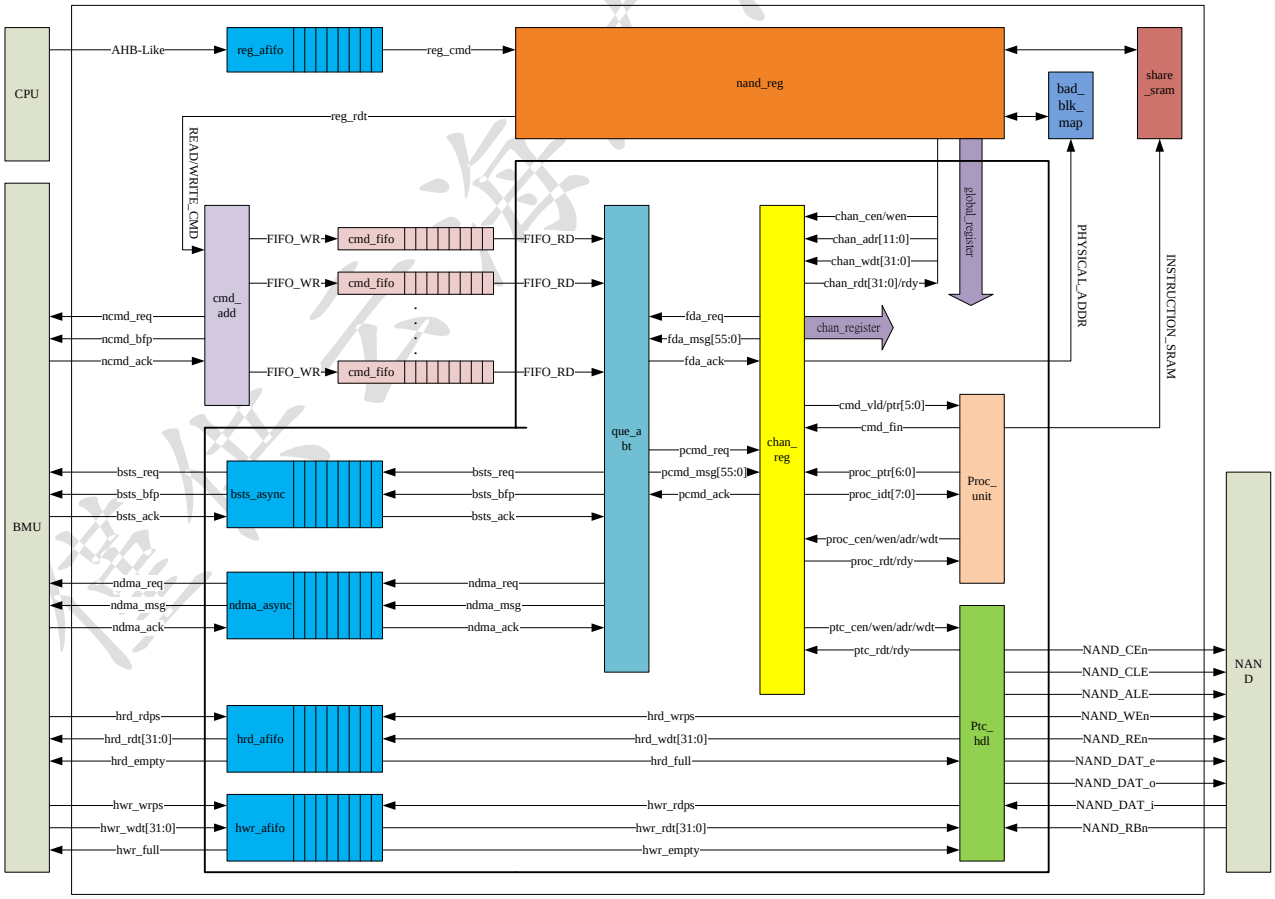


Figure 1. NCTL Architecture/Function Block

如上圖一所示, NCTL 主要是由 8 個 module 所組成. 每個 module 主要負責的 function 大致描述如下:

2.2. Component Description

3. INTERFACES AND PROTOCOLS

3.1. AHB-like Interface

Pin Name	Dir	Description

3.2. BMU Interface

Pin Name	Dir	Description

3.3. ONFI PHY Interface

Pin Name	Dir	Description
nand_cen[7:0]	Output	Chip Enable, 8bits (dfi_cebar[7:0])
nand_cle	Output	Command Latch Enable (dfi_cle)
nand_ale	Output	Address Latch Enable (dfi_ale)
nand_wen	Output	Write Enable (dfi_webar)
nand_ren	Output	Read Enable / reference clock (dfi_rd_pre_post_amble)
nand_rent	Output	Read Enable toggle (dfi_rddata_en, ~dfi_rebar)
nand_dqse	Output	Data strobe output enable (dfi_wrdqs_en)
nand_dqso	Output	Data strobe data (~dfi_sdrval_modifier)
nand_dqst	Output	Data strobe toggle (floating)
nand_dqe	Output	Data bus output enable (floating)
nand_dqo[15:0]	Output	Output data, 16bits (dfi_wrdata[15:0])
nand_dqv	Input	Input data valid, for DDR mode (dfi_rddata_valid)
nand_dqi[15:0]	Input	Input data, 16bits, for DDR mode (dfi_rddata[15:0])
nand_dql[7:0]	Input	Input data, 8bits, for legacy mode (pad_mem_data[7:0])
nand_rbn[7:0]	Input	Ready/Busy pin, 8bits (dfi_rbn[7:0])
dfi_wrdata_en	Output	(nand_cle || nand_ale) ? nand_dqe : nand_dqst
dfi_sdr_cycle	Output	data_state ? 1'b0 : (cmd_state ? 1'b1 : dfi_sdr_cycle)

4. PROGRAMMING MODEL

4.1. Control and Status Registers List

****特別注意，凡是 Type 是 RO，則代表 Read Only，因為並不存在實際的 register，所以並無 default value，讀到的值與當時的 status 有關. Type 是 RWC 則代表此 register 除了可以 Read/Write 外，H/W 會自動 Clear，也就是清成 0. W1C 則是代表寫 1，H/W 會清成 0**

Register Name	**CMD_INFO_Reg**
Register Address	*0x00~0xFF*

Osft	**Type**	**Bit**	**Def**	**Description**
0x0	RW	8	0	[5:0]: reg_cmd_ptr: Command pointer, this register map to micro code location. when write this register, imply to send this command to cmd queue.
0x1	RW	8	0	Queue info: [0]: reg_cmd_pri: priority queue [2:1]: reserved [3]: reg_cmd_end: command end / program unit end, HW can auto set this signal, see 0x28 register. [5:4]: reg_cmd_dmap: LBA to PBA direct mapping address, for 16TB/32TB. [7:6]: reserved
0x2	RW	16	0	reg_proc_info: FW send info to micro processor, reserved for future function
0x4	RW	32	0	reg_cmd_padr: NAND Physical Addr, {Blk uint, Page, CE, Chan, Frag}, if the capacity is 16TB/32TB, H/W would auto insert 0x1[5:4] into reg_cmd_padr
0x8	RW	12	0	reg_bmsg_bfp: BMU buffer pointer in buffer message. The buffer pointer is based on MAPU as a unit 0x0~0x9ff: 2560MAPU, DRAM buffer range 0xa00~0xbff: 512MAPU, SRAM buffer range
0xa	RW	6	0	reg_bmsg_msz: meta data size in buffer message. [2:0]: host meta size (0:8Bytes, 1:16Bytes, 2:32Bytes, 3:64Bytes, 4:128Bytes, 5:0Bytes) [5:3]: FW meta size (0:4Bytes, 1:8Bytes, 2:12Bytes, 3:16Bytes,

				4:20Bytes, 5:24Bytes, 6:28Bytes, 7:32Bytes)
0xb	RW	7	0	reg_bmsg_psz: parity data size in buffer message. See ECC design spec
0xc	RW	5	0	reg_bmsg_oth: [1:0]: reg_bmsg_fci: force ECC port info: bit0:enable, bit1: port number [4:2]: reg_bmsg_que: ECC decoder output to BMU queue number, 0:to Host queue, 1: to DRAM queue, 2: to CPU queue, 7:Non 4KB
0xd	RW	7	0	reg_bmsg_rid: RAID table index
0xe	RW	8	0	reg_bmsg_lid: LDPC index
0xf	RW	1	0	reg_bmsg_ltp: LDPC index type: 0:input buffer ptr, 1: LDPC table ptr
0x10	RW	8	0	reg_bmsg_ofs0: host cmd 4KB offset
0x12	RW	10	0	reg_bmsg_tag0: host cmd tag
0x14	RW	8	0	reg_bmsg_ofs1: host cmd 4KB offset
0x16	RW	10	0	reg_bmsg_tag1: host cmd tag
0x18	RW	8	0	reg_bmsg_ofs2: host cmd 4KB offset
0x1a	RW	10	0	reg_bmsg_tag2: host cmd tag
0x1c	RW	8	0	reg_bmsg_ofs3: host cmd 4KB offset
0x1e	RW	10	0	reg_bmsg_tag3: host cmd tag
0x20	RW	5	0	[0]: reg_fda_req: enable dma, H/W would auto clear this bit [1]: reg_fda_nwr: 1: write dma, 0: read dma [4:2]: reg_fda_chan: dma channel number
0x21	RO	2	0	
0x24	RW	8	0 0 0 0 0 0 0 1	[0]: reg_qswh_en: queue switch enable [1]: reg_rcah_en: read cache mode enable [2]: reg_rcah_allq: read cache mode for two priority queue [3]: reg_wcah_en: write cache mode enable [4]: reg_aipr_en: AIPR mode enable [5]: reg_susp_en: program/erase suspend enable [6]: reg_wseq_en: [7]: reg_bsts_chk
0x25	RW	8	0 0	[5:0]: reg_dmap_sft: insert cmd_dmap location [7:6]: reg_dmap_bit: insert cmd_dmap bit number

0x26	RW	6	4	reg_rdt_cptr: cmd_ptr[5:0] for read data
0x27	RW	6	5	reg_rcah_cptr: cmd_ptr[5:0] for cache read end
0x28	RW	6	0	reg_frag_end: 4KB number in program unit, 0 base 0: 4KB number is calculate by reg_frag_sht 1: 4KB number is 2 2: $KB number is 3 …… for example, if 6plane, 16KB page, the 4KB number is 6plane x 16KB / 4KB = 24, so set reg_frag_end = 23
0x29	RW	3	111	[0]: reg_auto_wrend: enable auto write cmd end function, HW would auto set reg_cmd_end(0x1[3]) [1]: reg_auto_rdend: enable auto read cmd end function, HW would auto set reg_cmd_end(0x1[3]) [2]: reg_cmd_to: auto cmd time out enable
0x2c	RW	32	4096	reg_to_thr: timeout threshold, the unit is clock cycle
0x30	RW	64	0110	reg_cmd_tp0: 64bits mapping to cmd_ptr[5:0] 0: read cmd, other cmd 1: program cmd, erase cmd
0x38	RW	64	1100	reg_cmd_tp1: 64bits mapping to cmd_ptr[5:0] 0: non data cmd type: erase cmd, other cmd(read status cmd, set feature cmd…..) 1: data cmd type: read cmd, program cmd
0x40	RW	8	0	reg_qcnt_sel: queue cmd cnt select
0x42	RO	12		que_cnt: queue cmd cnt

Register Name			**NAND_Set_Reg**	
Register Address			*0x100~0x1FF*	
Osft	**Type**	**Bit**	**Def**	**Description**
0x0	RW	4	0	reg_index [3] = 0 : [2:0] is channel select [3] = 1 : only for write, write all channel
0x1	RW	2	1	reg_sram_dly: SRAM read data delay cycle, only implement delay 1

				cycle, so must set as 2'd1
0x2	RW	2	01	reg_nand_md: NAND interface mode [0]: reg_legacy_md: NAND interface Legacy mode, before change this register, need set nand_cen = 0xff [1]: reg_cadence_md: If ONFI mode & ASIC, set 1, for FPGA, set 0
0x3	RW	2	0	reg_inst_share: Micro processor instruction sram share mode 0: no share 1: ch0~1 share the same instruction SRAM 2: ch0~3 share the same instruction SRAM 3: ch0~7 share the same instruction SRAM
0x4	RW	3	3	reg_frag_sft: frag number setting, frag imply 4KB number in read unit 0: 1 frag 1: 2 frag 2: 3~4 frag 3: 5~8 frag 4: 9~16 frag 5: 17~32 frag 6: 33~64 frag
0x5	RW	2	3	reg_chan_sft: channel number setting 0: 1 channel 1: 2 channel 2: 3~4 channel 3: 5~8 channel
0x6	RW	3	3	reg_ce_sft: CE number setting
0x7	RW	3	3	reg_lunce_sft: {LUN, CE} number setting
0x8	RW	4	12	reg_pg_sft: Page number setting
0x9	Rw	4	9	reg_blk_sft: Block number setting
0xa	RW	3	1	reg_plane_sft: plane number setting
0xb	RW	5	2	reg_plane_num: plane number
0xc	RW	13	4640	reg_mapu_size : Code Word(Mapping unit) size, (4096Bytes + Meta data + ECC parity) Bytes
0xe	RW	16	18560	reg_pg_size: NAND page size, Micron B27B is 18560Bytes, B58R is 18352Bytes

0x10	RO	64		Reserved
0x18	RW	64		reg_phy_lunce: logic bank mapping to physical LUNCE table
			0	[3:0]: logic bank0 mapping to physical LUNCE number
			1	[7:4]: logic bank1 mapping to physical LUNCE number
				……
			15	[63:60]: logic bank15 mapping to physical LUNCE number
0x20	RW	256		reg_frag_plane: frag number mapping to plane number table
			0	[3:0]: logic frag0 mapping to plane number
			0	[7:4]: logic frag1 mapping to plane number
			0	[11:8]: logic frag2 mapping to plane number
			0	[15:12]: logic frag3 mapping to plane number
			1	[19:16]: logic frag4 mapping to plane number
			1	[23:20]: logic frag5 mapping to plane number
				……
			15	[255:252]: logic frag63 mapping to plane number
0x40	RW	2		reg_sts_cfg: Nand read status config mode
			0	[0]: multi LUN or AIPR, need read status
			1	[1]: nand status from nand RBn pin
				If implement RBn pin, single LUN & disable AIPR, set 0x10, only use RBn pin to check whether NAND ready
				If implement RBn pin, multi LUN or AIPR, set 0x11, use timer or RBn to check whether need to read status
				If no RB pin, set 0x00, use timer to check whether need to read status
0x41	RW	6	0	reg_rds_cptr: Micro code cmd_ptr for read status
0x44	RW	8	0x60	reg_rds_mask: The bit mask for NAND ready/busy status
0x45	RW	8	0x60	reg_rds_info: The bit data for NAND ready (status_data & reg_rds_mask) = reg_rds_info, imply NAND status ready
0x46	RW	8	0x03	reg_fail_mask: The bit mask for NAND program/erase fail/ok status
0x47	RW	8	0x0	reg_fail_info: The bit data for NAND program/erase ok (status_data & reg_fail_mask) = reg_fail_info, imply NAND program/erase ok
0x48	RW	16	600	reg_tmr_base: timer unit (us), if system clock is 600MHz, set 600

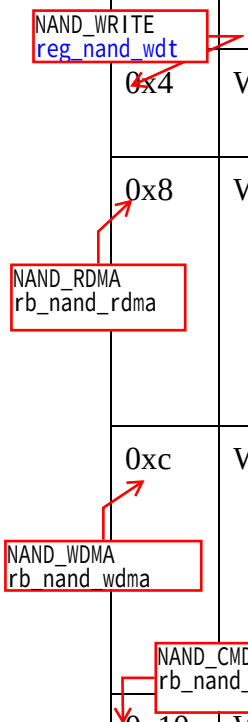
0x4c	RO	8		sts_fail: Channel map for program/erase fail
0x4d	RW	8	0xff	reg_sts_mask: Interrupt mask for program/erase fail
0x50	RW	128		reg_log_rbn: nand RBn pin mapping to logic RBn bit map table
			0x01	[15:0]: physical RBn0 pin mapping to logic RBn bit map
			0x02	[31:16]: physical RBn1 pin mapping to logic RBn bit map
			0x04	[47:32]: physical RBn2 pin mapping to logic RBn bit map
				…….
			0x80	[127:112]: physical RBn7 pin mapping to logic RBn bit map
0x60	RW	256		reg_tmr_dat: NAND busy time threshold, when use timer mode to check whether need to read status, set this threshold register
			0x0	[15:0]: NAND busy timer threshold 0
			0x0	[31:16]: NAND busy timer threshold 1
				……..
			0x0	[255:240]: NAND busy timer threshold 1
0x80				BBRM(Bad Block ReMapping) setting & status
	RWC	1	0	[0]: reg_bbt_req: BBRM build table process enable, HW auto clear
	RW	1	0	[1]: reg_bbt_clr: 0: add entry, 1: clear entry
	RO	1		[2]: bbt_err: Add entry error, reg_bbt_pblk = null
	RO	1		[3]: bbt_fail: add entry fail
	RO	1		[4]: bbt_owr: add entry over write
	RO	1		[5]: bbt_rpt: add entry repeat
	RO	1		[6]: bbt_noh: clear entry not hit
	RO	1		[7]: bbt_done: BBRM build table process done. When enable next “BBRM build table process”, HW would auto clear this bit. When BBRM build table process done, HW would set this bit as 1.
0x81	RW	7	0	reg_bbt_die: BBRM input die number [3:0]: Lun/CE number [6:4]: channel number
0x82	RW	14	0	reg_bbt_lblk: BBRM input logical block number
0x84	RW	14	0	reg_bbt_pblk: BBRM remapping physical block number
0x86	RO	13		bbt_entry: BBRM hit entry number
0x88	RW	5	0	reg_ext_sel: Select register entry

0x89	RW	7	0	reg_ext_die: die number of register entry [3:0]: Lun/CE number [6:4]: channel number
0x8a	RW	14	0	reg_ext_lblk: logical block number of register entry
0x8c	RW	14	0x3f f	reg_ext_pblk: physical block number of register entry, 0x3ff imply null
0x90	RW	104	 31 31 31 31 14 13	reg_bbrm_hash: BBRM hash function select bit, set 31 imply null [4:0]: BBRM bit0 select of hash function [12:8]: BBRM bit1 select of hash function [20:16]: BBRM bit2 select of hash function [28:24]: BBRM bit3 select of hash function ……. [92:88]: BBRM bit11 select of hash function [100:96]: BBRM bit12 select of hash function
0xa0	RW	128	 0 0 0	reg_dma_len: NAND read/write DMA length [15:0]: dma length 0 [31:16]: dma length 1 ……. [127:112]: dma length 7
0xc0	RW	8	2	tCR, tCR2: please reference Micron B47R NAND datasheet
0xc1	RW	8	29	tCS
0xc2	RW	8	2	tCS1, tCS2
0xc3	RW	8	2	tCD
0xc4	RW	8	2	tCSD
0xc5	RW	8	2	tCEH
0xc6	RW	8	2	tCLR
0xc7	RW	8	2	tAR
0xc8	RW	8	2	tCALS2
0xc9	RW	8	47	tWHR
0xca	RW	8	39	tWC
0xcb	RW	8	19	tWH
0xcc	RW	8	2	tCH
0xcd	RW	8	2	tADL

0xce	RW	8	80	tRHW
0xcf	RW	8	2	tCDQSS
0xd0	RW	8	2	tRPRE
0xd1	RW	8	2	tWPRE
0xd2	RW	8	19	tCALS
0xd3	RW	8	7	tCALH
0xd4	RW	8	19	tWP
0xd5	RW	8	2	tCAS
0xd6	RW	8	2	tDQSRH
0xd7	RW	8	2	tRPST
0xd8	RW	8	2	tRPSTH
0xd9	RW	8	2	tCHZ
0xda	RW	8	2	tCLHZ
0xdb	RW	8	2	tWPST
0xdc	RW	8	2	tWPSTH
0xdd	RW	8	2	tCDQSH
0xde	RW	8	2	tDBS
0xdf	RW	1	0	Enable tDBS timing constraint
		1	1	Enable tRHW timing constraint
		1	1	Enable tWPSTH timing constraint
		1	1	Set tWPSTH/tRPSTH timing condition
0xe0	RW	8	20	tLRS: legacy read setup time
0xe1	RW	8	20	tLRH: legacy read hold time
0xe2	RW	8	20	tLWS: legacy write setup time
0xe3	RW	8	20	tLWH: legacy write hold time
0xe4	RW	8	18	reg_lrd_dly: Legacy mode read latch time
0xf0	RW	256		Cadence PHY control register

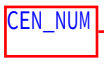
Register Name	**Channel_Set_Reg**			
Register Address	*0x1000~0x10FF*			
sft	**Type**	**Bit**	**Def**	**Description**
0x0	WO	16	0	reg_nand_rdt: Read data from NAND interface

NAND_READ
rw_nand_rt



				If write this register(any value), ptc_hdr would read data from NAND to 0x1058[15:0]
0x4	WO	16	0	reg_nand_wdt : Write data to NAND interface If write this register, ptc_hdr would write data to NAND
0x8	WO	8	0	reg_ena_rdma: Enable H/W Read DMA engine [7]=0, [6:0]=0, use reg_cmd_mapu(0x1041) as ECC unit number [7]=0, [6:0]!=0, use [6:0] as ECC unit number [7]=1 : [2:0] mean reg select, use reg_dma_len(0x1a0) as DMA length
0xc	WO	8	0	reg_ena_wdma: Enable H/W Write DMA engine [7]=0, [6:0]=0, use reg_cmd_mapu(0x1041) as ECC unit number [7]=0, [6:0]!=0, use [6:0] as ECC unit number [7]=1 : [2:0] mean reg select, use reg_dma_len(0x1a0) as DMA length
0x10	WO	8	0	reg_nand_cmd: Send Command to NAND interface
0x14	WO	8	0	reg_nand_adr: Send Address to NAND interface
0x18	WO	8	0	reg_ena_cen: Enable/Disable CEn: [7:4]!=0: disable all CE [7:4]=0: [3:0] mean CE number, when enable CE, auto disable all other CE
0x1c	WO	8	0	reg_ena_wait: Enable wait function: [7:0] mean wait clock cycle
0x20	RO	256	0	reg_proc_r: micro processor internal register [7:0]: R0 [15:8]: R1 ……. [255:248]: R31
0x40	RW	6	0	reg_cmd_ptr: read_cmd, read_data, prog, erase….. When write this register, imply micro processor would execute this command
0x41	RW	7	0	reg_cmd_mapu: DMA mapu number(4KB number), reg_cmd_eccu=1~64
0x42	RW	16	0	reg_proc_info: CPU send information to micro processor

0x44	RW	29	0	reg_cmd_blkpg: {NAND Physical Blk number, Page Number}, before bad block remapping
0x48	RW	2	0	reg_cmd_ect: H/W send info to micro processor [0]: NAND read cache cmd [1]: NAND write cache cmd
0x49	RW	6	0	reg_cmd_frag: mapu offset in program unit
0x4a	RW	4	0	reg_cmd_lunce: {LUN number, CE number}
0x4b	RW	3	0	chan_loc: current channel
0x4c	RO	4		phy_ce: Current command CE after remapping
0x4d	RO	3		phy_lun: Current command LUN after remapping
0x4e	RO	1		single_plane: one plane command
0x4f	RO	2	0x01	die_sts [0]: NAND die ready, H/W use lunce to select the NAND ready [1]: NAND die program/erase fail, H/W use lunce to select the NAND die
0x50	RO	4	0	plane_cnt: NAND DMA current plane number
0x51	RO	6	0	mapu_cnt: NAND DMA transfer mapu count
0x52	RO	16	0	col_cnt: NAND DMA current column address in page
0x54	RO	29		row_addr: {Lun_num, Bad block remapping blk, current_plane, page_num}, H/W would auto calculate plane number
0x58	RO	32		nand_rdata: NAND interface read data, when write 0x1000 (reg_nand_rdt), H/W would read data from NAND, and put the data to this register.
0x5c	RWC	16	ffff	status_rdy: Die map for NAND ready, write 1 to set
0x5e	RWC	16	0	status_fail: Die map for NAND program/erase fail, write 1 to clear
0x60	RW	6	0	[3:0]: reg_tmr_sel: Select 0x160~0x17F nand busy time threshold [4]: reg_frc_rds: For program/erase, force read status [5]: reg_busy_ps: write only, When write 1, H/W set nand status busy
0x61	RW	8	0	reg_rds_dat: NAND Read status data, When write this register, H/W would check the value and set status_rdy & status_fail
0x62	RW	5	0	[3:0]: reg_plane_cnt: change row_addr(0x54) plane value [4]: reg_plane_sel: 0: use reg_plane_cnt for row_addr(0x54), 1:use plane_cnt(0x50) for row_addr(0x54)



0x63	RW	3	0	[0]: reg_bbrm_en: enable Bad Blk ReMap(bbrm) table [1]: reg_bbrm_exe: bbrm execution, when remap finish, H/W would set this bit as 0 [2]: reg_bbrm_rds: enable bbrm for read status command
0x64	RW RO	2	0	[0]: reg_dma_rdy: 0: don't return ready until H/W DMA done, 1: H/W return ready and FW need check "nand_dma_exe" [1]: nand_dma_exe: Read only, NAND DMA engine execution, when DMA done, H/W auto clear this bit
0x65	RW	1	0	reg_dma_pause: set NAND DMA engine to pause
0x66	RW	16	0	reg_jump_num: micro processor pc jump number
0x68	RW	5	0	[3:0]: reg_msg_sel: select reg_proc_r(R24 ~ R31), each lunce has an independent 8bytes register in R24~R31, total support 16 lunce [4]: reg_msg_auto: 1: use reg_cmd_lunce(0x4a) to select reg_proc_r, 0: use reg_msg_sel(0x70) to select reg_proc_r
0x69	RW	3	0	reg_func_except: [0]: reset micro processor [1]: reset nand interface function [2]: hold micro processor
0x6a	RW	6	0	reg_bus_keep: [0]: bus_keep state, nand_cle, nand_ale [1]: bus_keep state, nand_dqe, nand_dqo [2]: idle state, nand_cle, nand_ale [3]: idle state, nand_dqe, nand_dqo [4]: rwm_idle state, nand_ren [5]: rwm_idle state, nand_dqse, nand_dqso
0x6b	RW	8	0	reg_bus_mirr: NAND DQ bus mirror, different bit imply different CE
0x6c	RW	5	0x7	[3:0]: reg_lunce_end: lunce end, if 1 die, set 0, 2die, set 1…. [4]: reg_lunce_ps: initial queue arbiter function
0x6d	RW	5	0	reg_qptr_sel: select 0x84/0x86 queue pointer [3:0]: die select [4]: priority select
0x6e	RW	5	0	[3:0]: reg_die_pause: suspend execution of one die command [4]: reg_all_pause: suspend execution of all command

0x6f	RW	2	0	reg_que_pause [0]: low queue command pause [1]: high queue command pause
0x70	RW	32	0	[3:0]: reg_cind_wen: Byte write enable for indirect access [4]: reg_cind_req: request for indirect access, HW auto clear [27:8]: reg_cind_adr: address for indirect access
0x74	RW	32	0	reg_cind_dat: read/write data for indirect access
0x78	RO	15	0	inst_adr: micro processor program count location
0x7a	RO	5	0	ptc_mode: NAND cmd sequence state machine
0x7b	RO	4	0	nrw_mode: NAND onfi Read/Write state machine
0x7c	RO	5	0x8	[0]: cmd_act: micro processor active [1]: pcmd_req: send next command request [2]: nprd_full: NAND Read FIFO full signal [3]: npwr_empty: NAND write FIFO empty signal [4]: rd_miss: NAND's REn, DQS pulse number is not equal
0x7d	RW	2	1 1	[0]: dfi_rd_mode: param_extended_rd_mode: Cadence ONFI PHY read mode control register [1]: dfi_wr_mode: param_extended_wr_mode: Cadence ONFI PHY write mode control register
0x7e	RW	16	0	io_pad_cfg: NAND IO pad config register
0x80	RO	32	0	cmdq_vld [15:0]: low priority command queue valid, diff bit imply diff die [31:16]: high priority command queue valid, diff bit imply diff die
0x84	RO	12	0	que_rptr: command queue read pointer
0x86	RO	12	0	que_wptr: command queue write pointer
0x88	RO	32	all 1	que_rdy [15:0]: read command ready, diff bit imply diff die [31:16]: write command ready, diff bit imply diff die
0x8c	RO	3	0	que_abt_st: queue arbiter function state machine
0x8e	RO	8	0xff	nand_cen: ONFI PHY NAND CEn
0x8f	RO	8	0xff	nphy_rbn: ONFI PHY NAND RBn