FlexSPI Driver Design

FlexSPI controller is new IP from Microcontroller group and it will replace QSPI in all future SoCs.
FlexSPI is superset and superior to QSPI. Most of the feature set of FlexSPI and QSPI are same, but there are few difference related to IO signal width, command set, default LUT programming and Hyperflash support.
FlexSPI has AHB and IP bus interface. AHB 64-bit interface and is mainly use for READ and WRITE flash operation whereas IP is 32-bit interface and it supports all flash operation – READ, WRITE, STATUS CHECK, GET PARAMS etc.
FlexSPI programs various commands in LUT and these commands sequence are trigger when we do AHB/IP bus READ/WRITE operation.
This documents introduces FlexSPI controller, FlexSPI serial NOR driver implementation and FlexSPI serial NAND driver implementation.

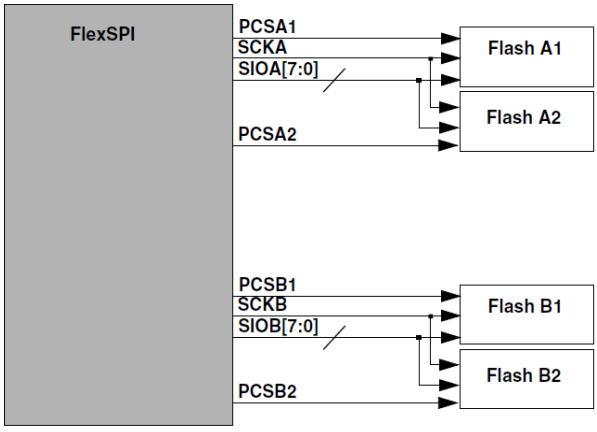
1. FlexSPI Controller Introdunction

FlexSPI is a flexible SPI host controller which supports TWO SPI channels and up to 4 external devices.
Each SPI channel support Single/Dual/Quad/Octal mode bi-directional data lines.
FlexSPI controller supports following memory devices

a) Serial NOR flash
b) Serial NAND flash
c) HyperBus devices (Hyperflash/HyperRAM)
d) FPGA devices

1.1 **FlexSPI Connections**

FlexSPI controller has two SPI channel and each channel can be connected to two devices. So, we can have 4 flash memory devices connected to FlexSPI controller at a time.
As shown in below connection diagram

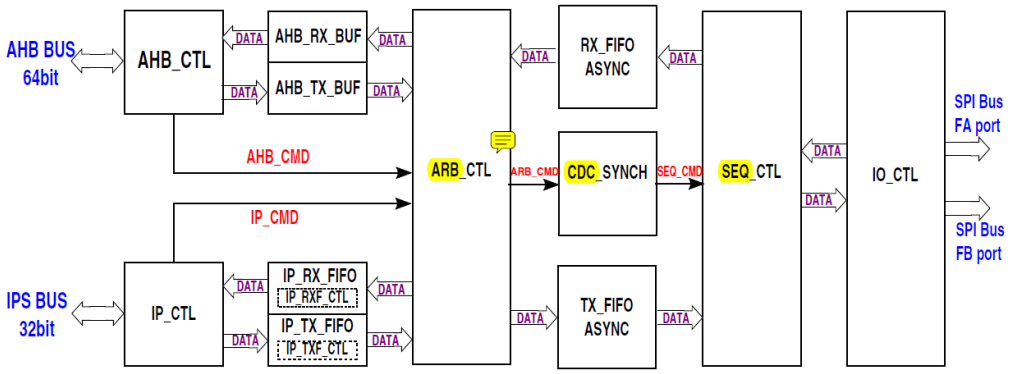


1.2 FlexSPI Command Interfaces

FlexSPI controller has two command interfaces

AHB Bus

IP CMD Bus



AHB command interface supports only two command – READ and WRITE to flash memory devices.

Whereas IP CMD interfaces support a lot of commands like – READ, WRITE, STATUS CHECK, GET PARAMS etc.

Another difference between AHB and IP CMD interface is that AHB is 64-bit interface whereas IP is 32-bit interface.

AHB command is normally used to access serial Flash memory space. IP command should be used to access the control and status registers in external flash device.

Flash could be accessed by AHB bus directly on AHB address space:0~10000000. This address space is mapped to Serial Flash Memory in FlexSPI. AHB bus access to this address space may trigger Flash access command sequence as needed.

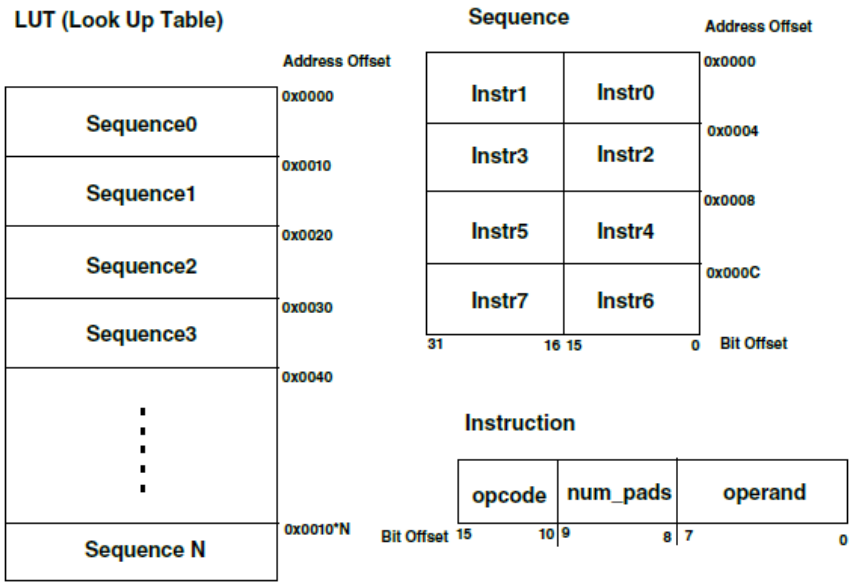
For AHB read access to Serial Flash Memory, FlexSPI will fetch data from SFM into AHB RX Buffers and then return the data on AHB Bus. For AHB write access to Serial Flash Memory, FlexSPI will buffer AHB Bus write data into AHB TX Buffer and then transmit to Serial Flash memory.

There is no software configuration or polling need for AHB command except FlexSPI initialization. AHB master access external flash device transparently similar as normal AHB slave.

1.3 FlexSPI Look Up Table(LUT)

The LUT (Look Up Table) is an internal memory to preserve a number of pre-programmed command sequences. Each sequence consists of 8 (2 byte each) instructions which are executed sequentially. When flash access is triggered by IP command or AHB command, FlexSPI controller will fetch the sequence from LUT according to sequence index/number and execute it to generate a valid flash transaction on FlexSPI interface

LUT and sequence structure



Each instruction is of TWO bytes containing OPCODE, PADS and OPERAND information. And a sequence is set of 8 x 2-byte instruction. If valid instruction in a LUT sequence are less than 8, then STOP command should be program for invalid instruction.

The reset value of LUT is unknown because it is implemented as internal memory. LUT should be programmed according to the device connected on board. To protect its contents during a code runover the LUT could be locked/unlocked to avoid change by mistake after programmed. The key for locking or unlocking the LUT is **0x5AF05AF0.** The process for locking and unlocking the LUT is as follows:

Locking the LUT

1. Write the key **(0x5AF05AF0)** in to the LUT Key Register (LUTKEY)
2. Write 2b01 to the LUT Control Register field LOCK and UNLOCK (LUTCR)

immediately after above KEY register writing. LUT is not successfully locked if there is any other register write access to FlexSPI between these two write access.

Unlocking the LUT

1. Write the key **(0x5AF05AF0)** in to the LUT Key Register (LUTKEY)
2. Write 2b10 to the LUT Control Register field LOCK and UNLOCK (LUTCR)

immediately after above KEY register writing. LUT is not successfully unlocked if there is any other register write access to FlexSPI between these two write access.

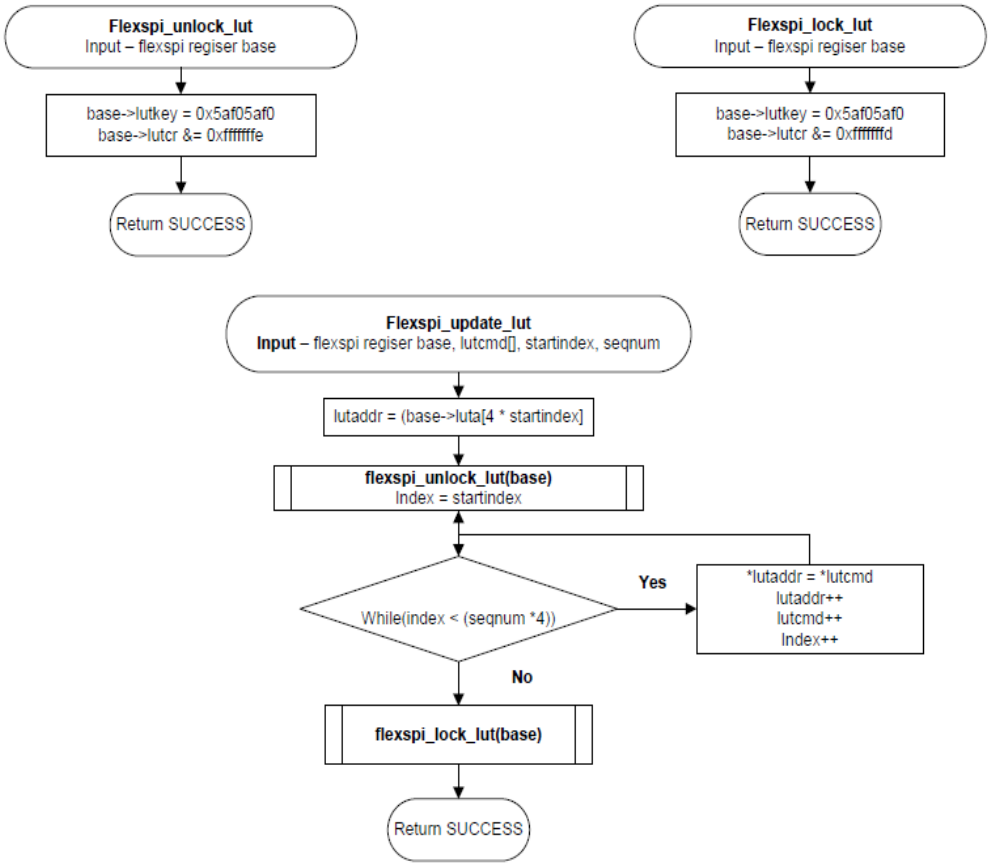
LUT Operations Flowcharts

There are mainly THREE operation we can perform on LUT –

(1) Unlock – LUT unlock required to write new instruction in LUT

(2) Lut_write/update – Write new instruction to look up table

(3) Lock – Lock LUT table to avoid unintentional writes.



1.4 FlexSPI Command Set (Programmable Sequence Engine)

FlexSPI controller implements a programmable sequence engine that execute the command sequence from LUT. FlexSPI controller executes the instructions sequentially and generate flash transaction on SPI interface accordingly. Following table is a list of supported instructions

Name	OPCODE	PADS	OPERAND
CMD_SDR	0x1	x1/x2/x4/x8	FLASH Command
RADDR_SDR	0x2	x1/x2/x4/x8	Row Address width (24-bit or 32-bit)
CADDR_SDR	0x3	x1/x2/x4/x8	Column address width
READ_SDR	0x9	x1/x2/x4/x8	Read data length
STOP	0x0	x1	Stop execution

The programmable sequence engine allows the software to configure the FlexSPI LUT according to external serial device connected on board. The flexible structure is easily adaptable to new command/protocol changes from different vendors

2 FlexSPI serial NOR driver implementation

Serial NOR flash memory can be access in different mode viz serial, dual, quad or octal (x1/x2/x4/x8).

FlexSPI serial NOR driver will implement mainly TWO functionalities for SP BootROM

(1) flexspi_nor_init –

This will initialize FlexSPI controller for AHB read access, program LUT with READ_SDR command sequence and initialize serial NOR memory (if required).

(2) flexspi_nor_read –

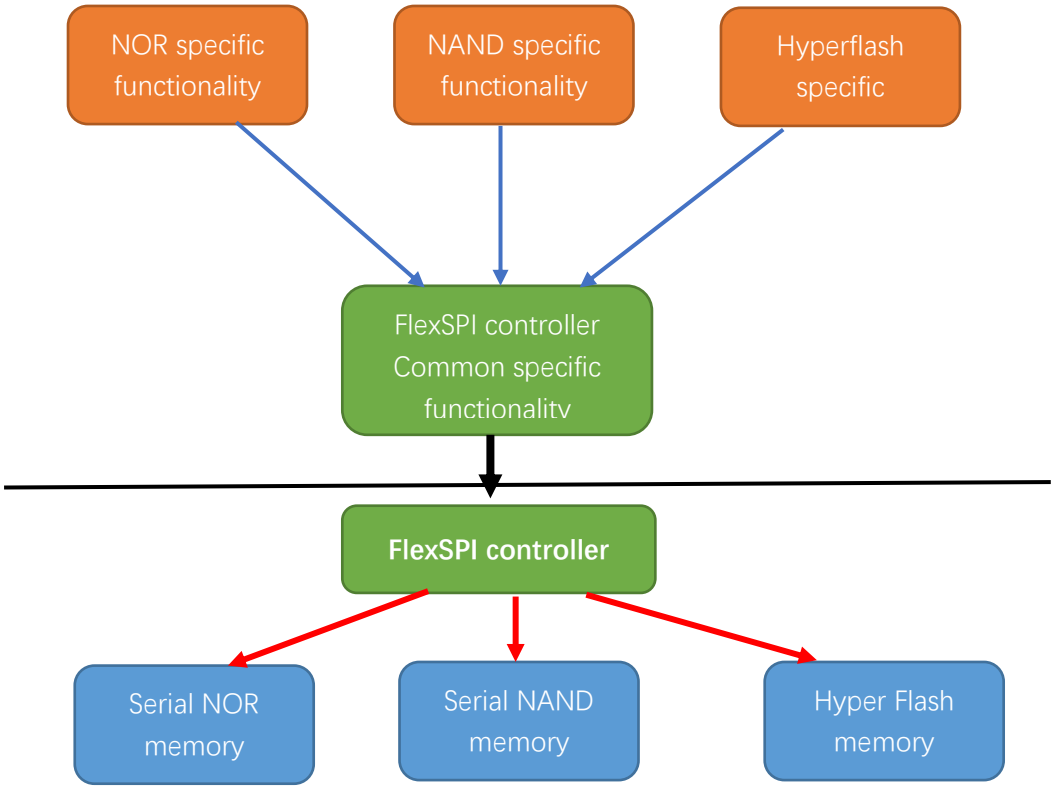
flexspi_nor_read will access serial nor memory from AHB interface and return read data to Service process (SP).

FlexSPI Driver File Hierarchy

FlexSPI driver has been divided into TWO parts –

Common functionality has been aggregated into files flexspi.c and flexspi.h

Serial NOR specific functionality is defined in file flexspi_nor.c and flexspi_nor.h



FlexSPI Serial NOR Data structures

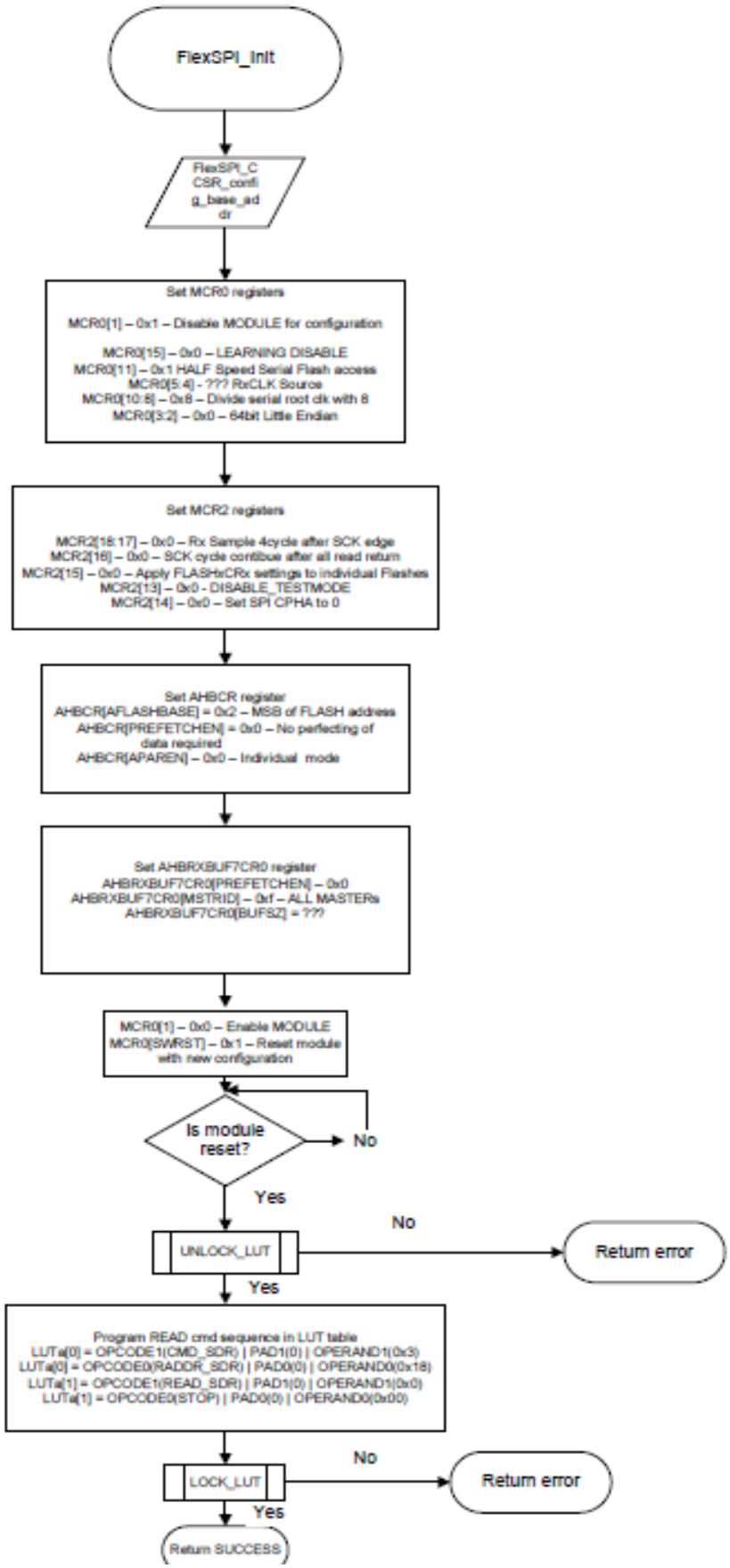
Data Structue	File Name	Description
struct flexspi_regs	flexspi.h	This structure defines all FlexSPI controller CCSR register map
enum cmds_seqid	flexspi.h	Define LUT table index (0…n) for various commands sequence
u32 nor_lut[]	flexspi_nor.c	This array defines command sequence programmed for Serial NOR

FlexSPI_INIT Function

Function Name	u32 flexspi_init()
Input Parameter	struct flexspi_regs *base
Description	This function initializes and set FlexSPI module, AHB bus and Flash Configuration Like - Set RxClk source, disable parallel mode, sw_reset etc

FlexSPI controller initialization sequence is as following:

1. Enable controller clocks (AHB clock/IP Bus clock/Serial root clock) in System level.
2. Set MCR0.MDIS to 0x1 (Make sure controller is configured in module stop mode)
3. Configure module control registers: MCR0, MCR1, MCR2. (Don't change MCR0.MDIS)
4. Configure AHB bus control register (AHBCR) and AHB RX Buffer control registers
5. (AHBRXBUFxCR) optionally if AHB command will be used
6. Configure Flash control registers (FLSHxCR0/1/2) according to external device type
7. Set MCR0.MDIS to 0x0 (Exit module stop mode)
8. Reset controller optionally (by set MCR0.SWRESET to 0x1)


FlexSPI_Init
FlexSPI_CCSR_config_base_addr
Set MCR0 registers
MCR0[1] – 0x1 – Disable MODULE for configuration
MCR0[15] – 0x0 – LEARNING DISABLE
MCR0[11] – 0x1 HALF Speed Serial Flash access
MCR0[5:4] - ??? RxCLK Source
MCR0[10:8] – 0x8 – Divide serial root clk with 8
MCR0[3:2] – 0x0 – 64bit Little Endian
Set MCR2 registers
MCR2[18:17] – 0x0 – Rx Sample 4cycle after SCK edge
MCR2[16] – 0x0 – SCK cycle contibue after all read return
MCR2[15] – 0x0 – Apply FLASHxCRx settings to individual Flashes
MCR2[13] – 0x0 - DISABLE_TESTMODE
MCR2[14] – 0x0 – Set SPI CPHA to 0
Set AHBCR register
AHBCR[AFLASHBASE] = 0x2 – MSB of FLASH address
AHBCR[PREFETCHEN] = 0x0 – No perfecting of data required
AHBCR[APAREN] – 0x0 – Individual mode
Set AHBRXBUF7CR0 register
AHBRXBUF7CR0[PREFETCHEN] – 0x0
AHBRXBUF7CR0[MSTRID] – 0xf – ALL MASTERs
AHBRXBUF7CR0[BUFSZ] = ???
MCR0[1] – 0x0 – Enable MODULE
MCR0[SWRST] – 0x1 – Reset module with new configuration
Is module reset?
No
Yes
UNLOCK_LUT
No
Return error
Yes
Program READ cmd sequence in LUT table
LUTa[0] = OPCODE1(CMD_SDR) | PAD1(0) | OPERAND1(0x3)
LUTa[0] = OPCODE0(RADDR_SDR) | PAD0(0) | OPERAND0(0x18)
LUTa[1] = OPCODE1(READ_SDR) | PAD1(0) | OPERAND1(0x0)
LUTa[1] = OPCODE0(STOP) | PAD0(0) | OPERAND0(0x00)
LOCK_LUT
No
Return error
Yes
Return SUCCESS

FlexSPI_NOR_INIT Function

Function Name	u32 flexspi_nor_init()
Input Parameter	void *p
Description	Initialize FlexSPI controller, program LUT table and initialize serial Flash for AHB read

Algorithm

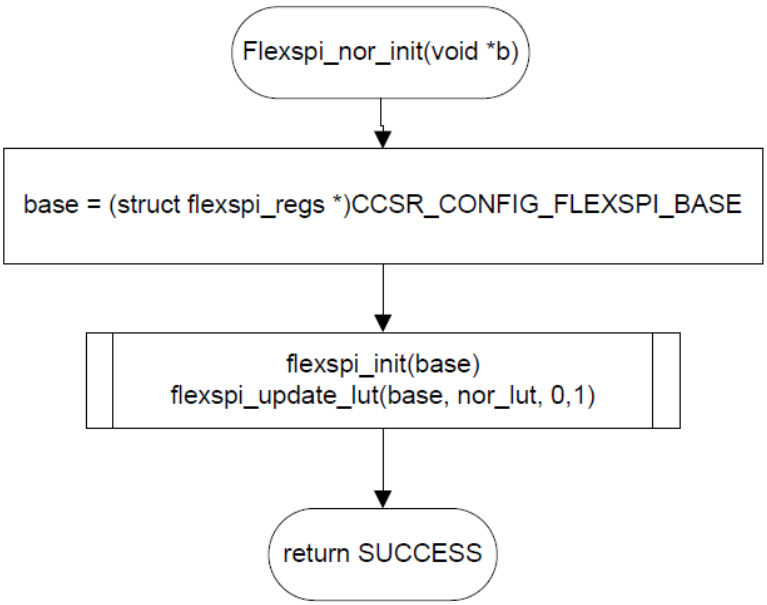
(1) Get FlexSPI CCSR register address in variable "**base**"

(2) Call flexspi_init(base)

(3) Call flexspi_update_lut(base, nor_lut, startindex, len)

(4) return SUCCESS

Flowchart



FlexSPI_NOR_READ Function

Function Name	u32 flexspi_nor_read()
Input Parameter	u32 src_offset u8 *dst u32 size void *x
Description	Read data via AHB bus from Serial NOR flash from src_offset to dst of len bytes

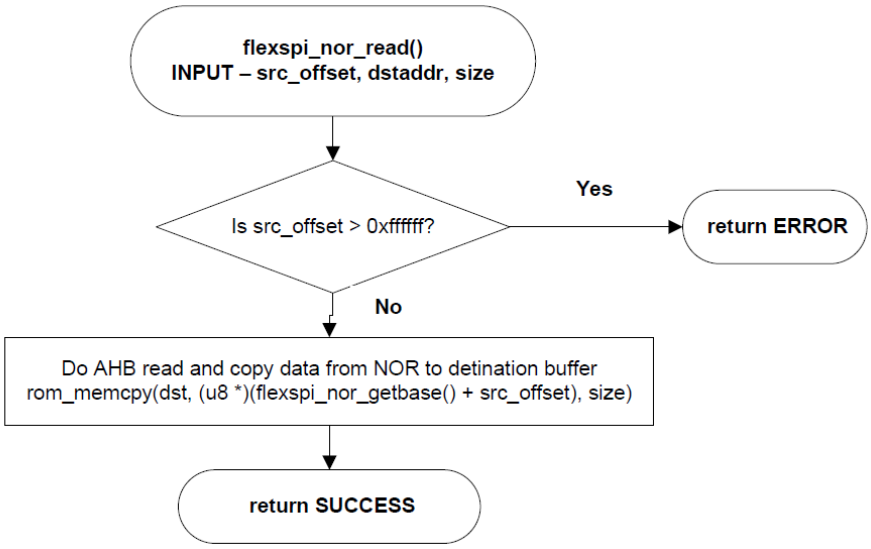
Algorithm

(1) Read only when src_offset < 0xFFFFFF

(2) rom_memcpy(dst, src, size)

(3) return SUCCESS

Flowchart



3 FlexSPI serial NAND driver implementation

Serial NAND flash memory can be access in different mode viz serial, dual or quad (x1/x2/x4). FlexSPI serial NAND driver will implement mainly TWO broad functions for SP BootROM

1. flexspi_nand_init –

   Flexspi_nand_init performance

   (1) FlexSPI controller initialization

   (2) Programs LUT table with various serial NAND command sequences

   (3) Get NAND parameters from serial NAND flash and get values for

   a. Page size

   b. Spare size

   c. Number of pages per block

   d. Is multiplane supported?

   (4) Bad block table initialization

2. flexspi_nand_read –

   flexspi_nor_read reads data from serial NAND flash. Following functionality has been implemented in the function

   (1) Read EVEN pages via AHB interface

   (2) Read ODD pages via IP command interface

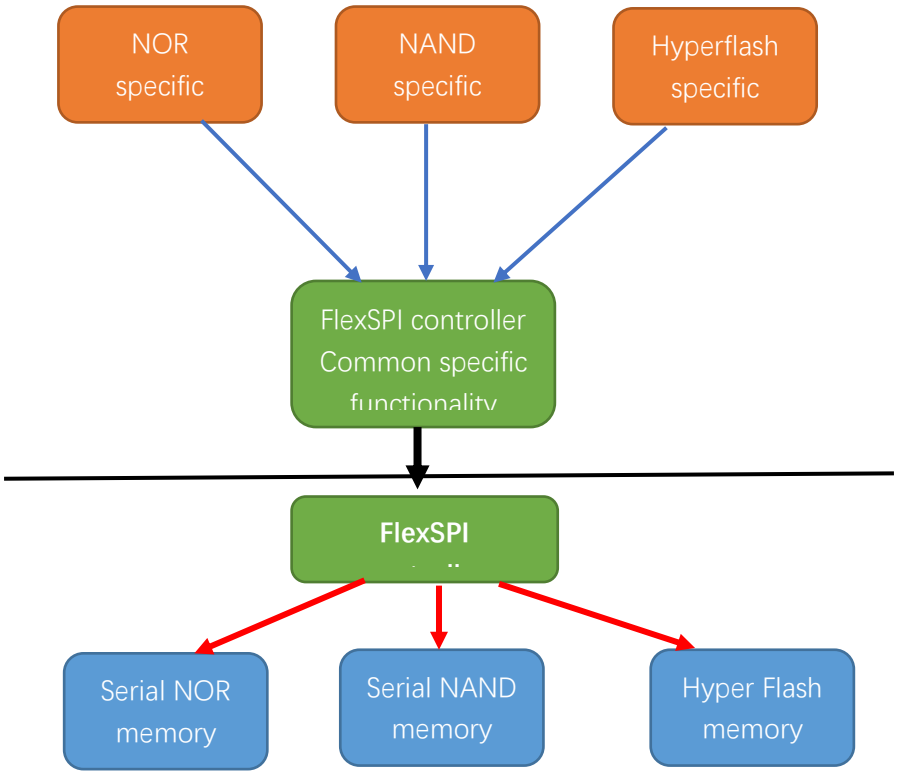
   (3) Does bad block handling

FlexSPI NAND Drivers File

As with FlexSPI NOR driver, we have divided FlexSPI NAND driver into TWO parts –

Common functionality has been aggregated into files flexspi.c and flexspi.h

Serial NOR specific functionality is defined in file flexspi_nand.c and flexspi_nand.h

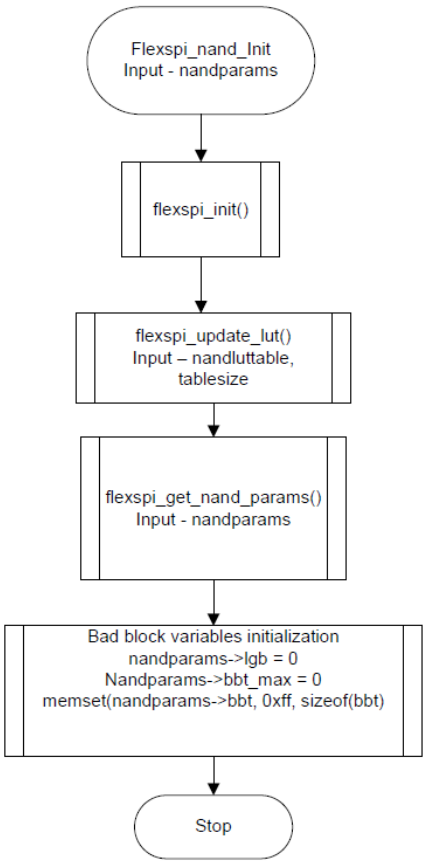


FlexSPI Serial NAND Data structures

Data Structue	File Name	Description
struct flexspi_regs	flexspi.h	This structure defines all FlexSPI controller CCSR register map
enum cmds_seqid	flexspi.h	Define LUT table index (0…n) for various commands sequence
Struct nandparams	Flexspi_nand.h	Define structure for storing serial NAND parameters e.g. pagesize, spare size, block size, multiplane etc
Struct nandinfo	Flexspi_nand.h	Store information about nand params, bad block table
u32 nand_lut[]	flexspi_nand.c	This array defines command sequence programmed for Serial NOR

FlexSPI_NAND_INIT Function

Function Name	u32 flexspi_nand_init()
Input Parameter	void *nandinfo
Description	This function initializes and set FlexSPI module, AHB bus and Flash Configuration, LUT programming, get paramaters and initialize bad block table



FlexSPI_NAND_READ Function

Function Name	u32 flexspi_nand_read()
Input Parameter	u32 src_addr, u8 *dst, u32 size, void *nandinfo
Description	This function read data from serial NAND memory

FlexSPI NAND read function basically does following

(1) Convert source address to block, current page number

(2) Read flash we SIZE bytes are read

a. Check for bad block, if block is bad find next good block

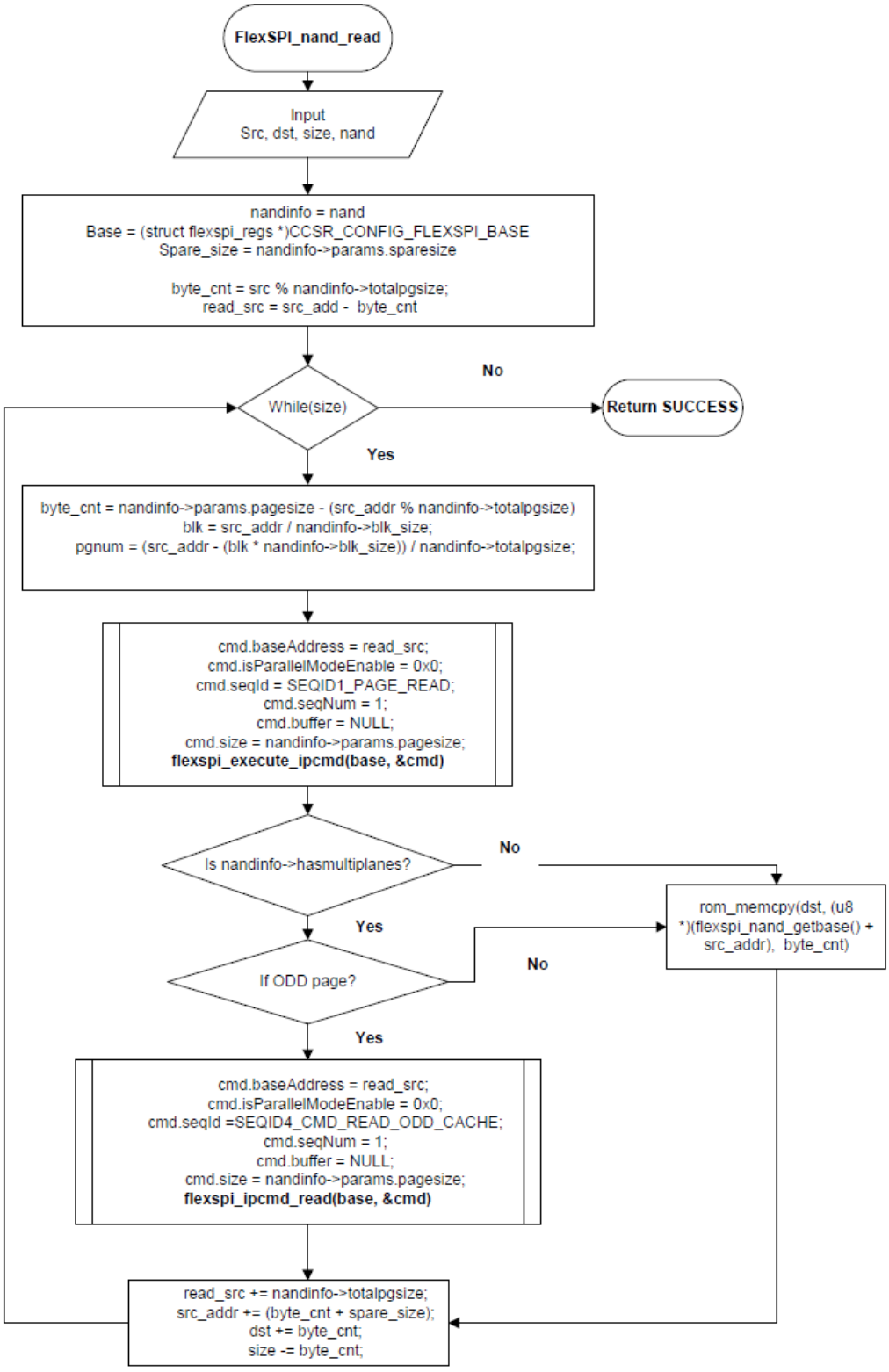
b. If multiplane enable

i. If pgnum is ODD, ready via IP CMD interface

ii. If pgnum is EVEN, ready via AHB interface

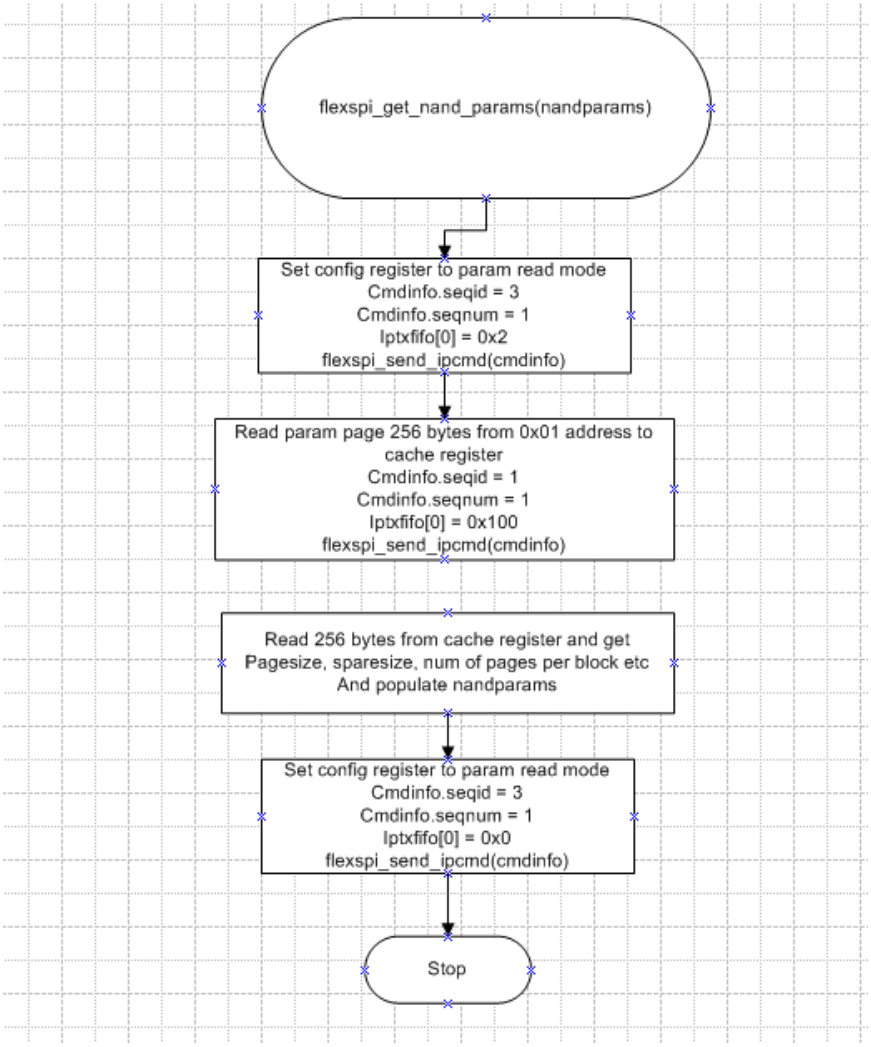
c. Read page from AHB interface

d. Increase page, decrease size by number of byte read



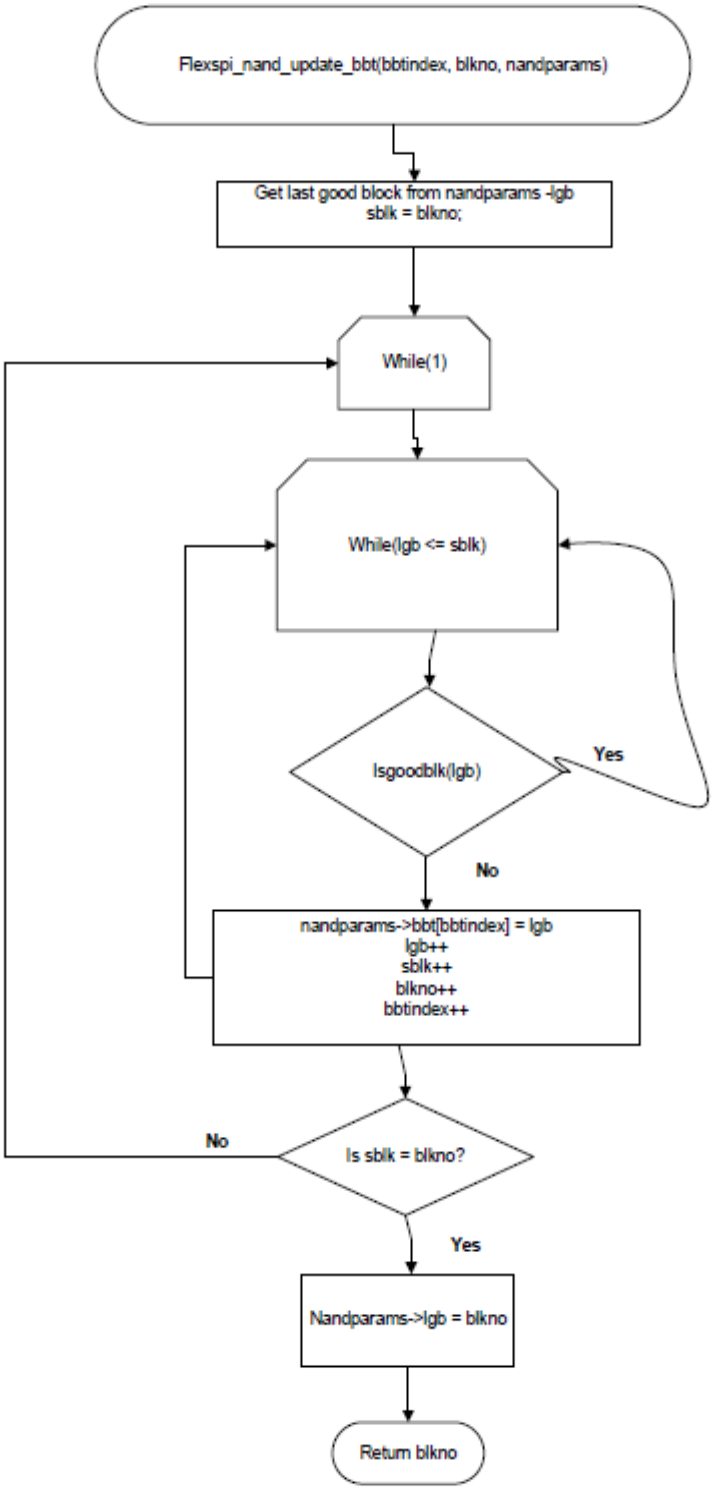
FlexSPI_get_nand_params

Function Name	u32 flexspi_get_nand_params()
Input Parameter	struct flexspi_regs *base, void *nandinfo
Description	Get serial NAND parameters from parameter page



Flexspi_badblock_mgmt

Function Name	u32 flexspi_update_bbt()
Input Parameter	U32 bbtindex, u32 blkno, void *nandinfo
Description	Check whether blkno is BAD or not.If YES, return GOOD block number


Flexspi_nand_update_bbt(bbtindex, blkno, nandparams)
Get last good block from nandparams -lgb
sblk = blkno;
While(1)
While(lgb <= sblk)
Isgoodblk(lgb)
Yes
No
nandparams->bbt[bbtindex] = lgb
lgb++
sblk++
blkno++
bbtindex++
Is sblk = blkno?
No
Yes
Nandparams->lgb = blkno
Return blkno