FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Introduction to Integrated Flash Controller

FTF-NET-F0109

Kevin Lam
Application Engineer

August 2012

Agenda

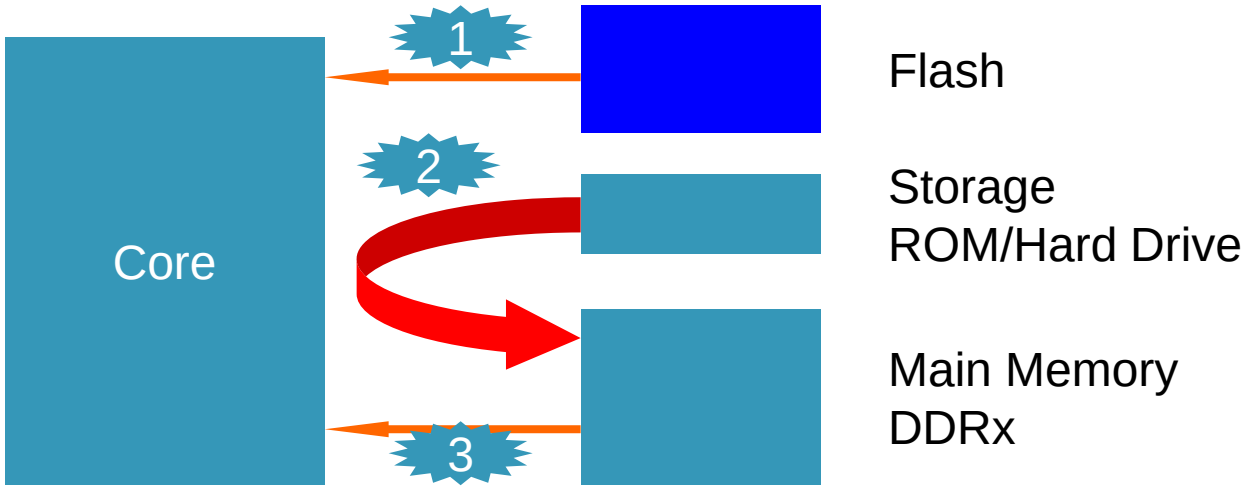
- Introduction to Integrated Flash Controller (IFC)
- Difference between eLBC and IFC
- IFC architecture
- NOR controller
- NAND controller
- GPCM controller

Usage of Integrated Flash Controller (IFC)

Role of IFC

After the power is turned on, a typical system:

1. Starts execution from non-volatile memory, typically NOR/NAND flash memory
2. Copies the code to main memory (DDRx)
3. Starts the execution from main memory



Initial Boot

- A system must be able to talk to non-volatile memory without any software configuration steps
- Integrated Flash Controller does this initial booting job.
  - IFC has three controllers:
    1. NOR controller
       - Standard and page mode NOR flash
       - **Support booting**
    2. NAND flash control machine (FCM)
       - NAND memory for storage
       - **Support booting**
    3. GPCM
       - Normal GPCM: Legacy, standard NOR flash
       - Generic ASIC: FPGA
       - **No booting**

Evolution of the local bus

- **MPC8xx Memory Controller (1995)**
  - GPCM: ROM, SRAM
  - UPM: DRAM

- **MPC82xx Memory Controller (1998)**
  - GPCM
  - UPM
  - SDRAM controller
  - Improvement: Dedicated SDRAM controller

- **MPC83xx/MPC85xx Local Bus Controller (2003/2004)**
  - GPCM
  - UPM
  - SDRAM controller
  - Improvement: Dedicated DDR controller supported outside of LBC

Evolution of the local bus

QorIQ P10xx/P20xx/P40xx eLBC(Enhanced LBC) (2008)

- GPCM
- UPM
- NAND Flash Control Machine
- Improvement: Removed SDRAM controller
  Added NAND controller (small page, 2K page)

QorIQ P1010 (IFC – Integrated Flash Controller rev1.0) (2011)

- GPCM (Normal GPCM, Generic ASIC)
- NOR
- NAND Flash Control Machine
- Improvement: Removed UPM
  Improved GPCM
  Added NOR controller
  Improved FCM, added 4K page (rev 1.0) support

eLBC, IFC Comparison

Machine	Features	IFC	eLBC
NAND Flash	Devices max page size	IFC rev1.0: 4KB IFC rev1.1: 8KB	2KB
	Error correction	IFC rev1.0: 4-bit, 8-bit/512Byte IFC rev1.1: 4-bit, 8-bit/512Byte 24-bit, 40-bit/1Kbyte	1-bit /512Bytes
	Flexible timing control allows interfacing with proprietary NAND	Yes	Limited capability
	Provide cache, copy-back and multi-plane command support	Yes	No
	Programmable command and data transfer sequences	Up to 15	Up to 8
	BBI page position	Configurable between (2^{nd} and last page)	First 2 pages of each block
	Configurable block size constraint to multiple of 32 pages, up to 2048 pages	Yes	No
	Internal SRAM size	9KB	5KB
	Max initial boot code size for NAND flash	8KB	4KB
	Internal SRAM access while NAND operation is on	Not-Allowed	Allowed

eLBC, IFC Comparison (continued)

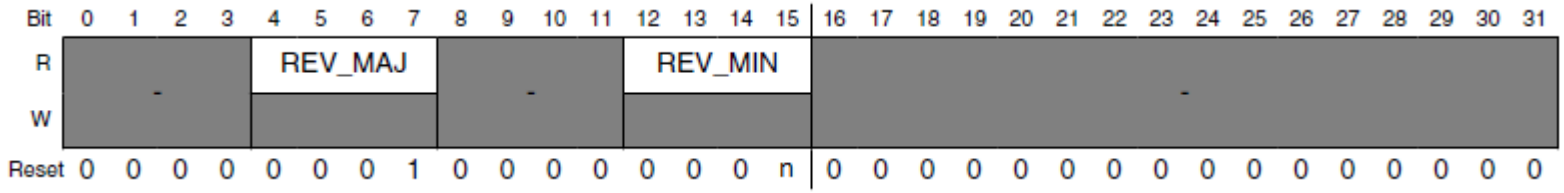
Machine	Features	IFC	eLBC
NOR Flash	Compatible with page mode NOR flash	Yes	No
	Flexible timing control allows interfacing with variety of NOR devices	Yes	Limited capability
	GPCM	Yes (with enhanced timing control)	Yes
Other	UPM	No	Yes
	GASIC	Yes	No

IFC Versions

- Currently two versions of IFC:

  rev 1.0, rev 1.1
- How to know the version number

  IFC revision control register (IFC_REV), offset 0x0



Rev1.0: P1010

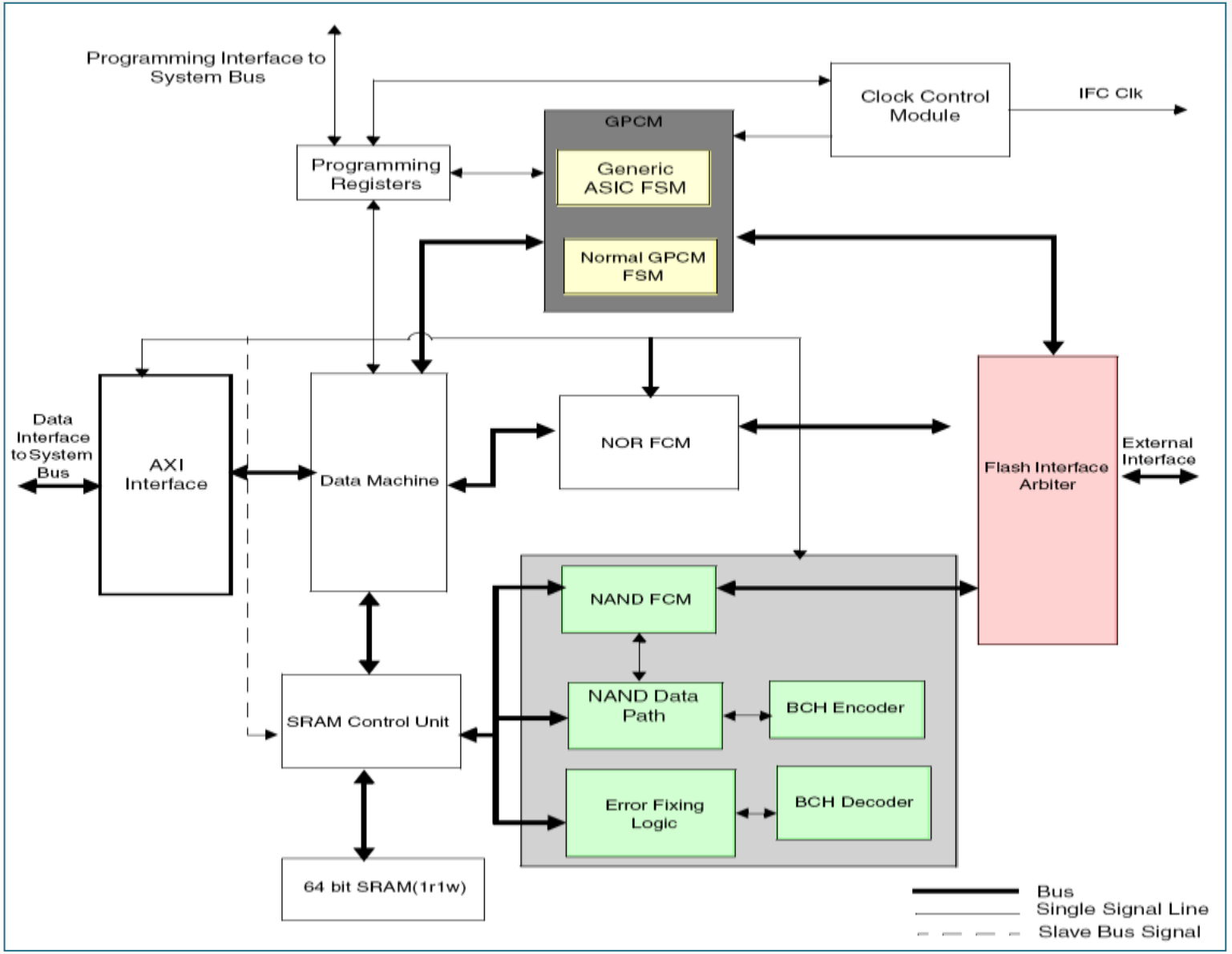
Rev1.1: T4240

- Some other improvement in rev1.1
  - Chip Select increases from 4 to 8.
  - Added support for External Transceiver Enable Control to support both slow and fast device in the local bus

Interface Signals

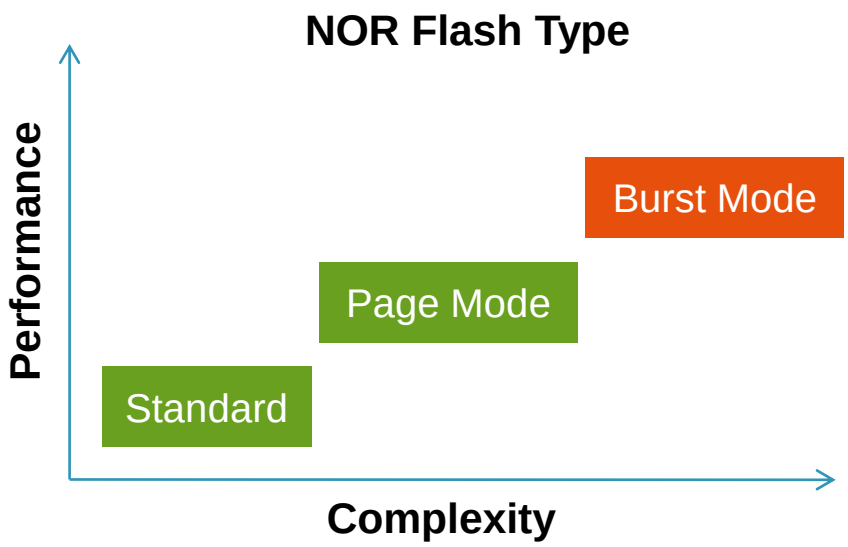
Name	IFC	eLBC
Address/Data	IFC_AD	LAD
Address	IFC_ADDR	LA
Address Valid	IFC_AVD	LALE/LFALE
Chip Select	IFC_CS_B	LCS
Write Enable	IFC_WE_B	LWE_B
Command Latch Enable	IFC_CLE	LFCLE
Output Enable	IFC_OE_B	LOE_B
Write Protect	IFC_WP_B	LFWP_B
Ready/Busy/Termination	IFC_RB_B/IFCTA_B	LFRB_B/LGTA_B
Buffer Control	IFC_BCTL	LBCTL
Parity	IFC_PAR[0:1]	LDP[0:1]
Parity Error	IFC_PERR	Not Available
Clock	IFC_CLK	LCLK

IFC Architecture

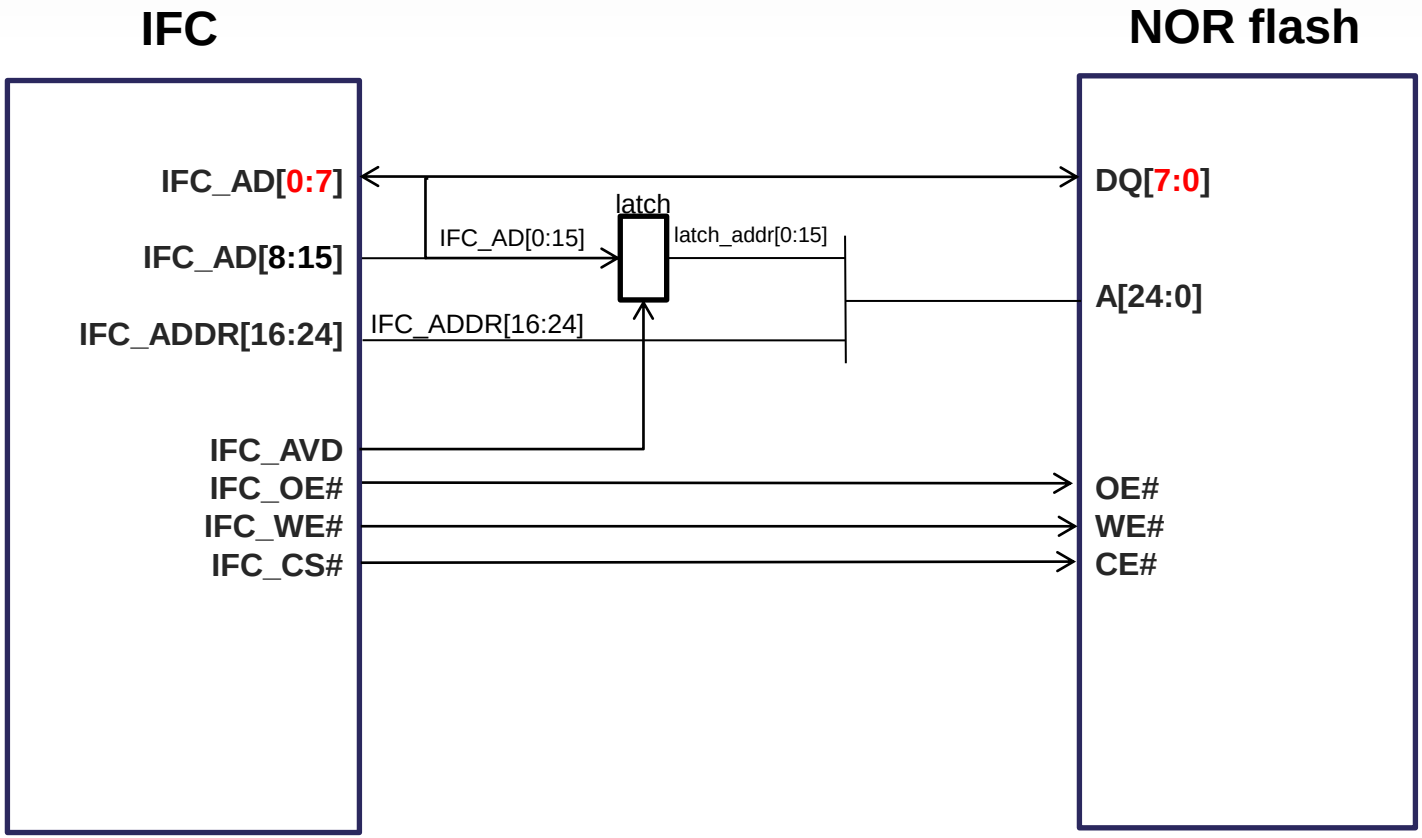


NOR Controller

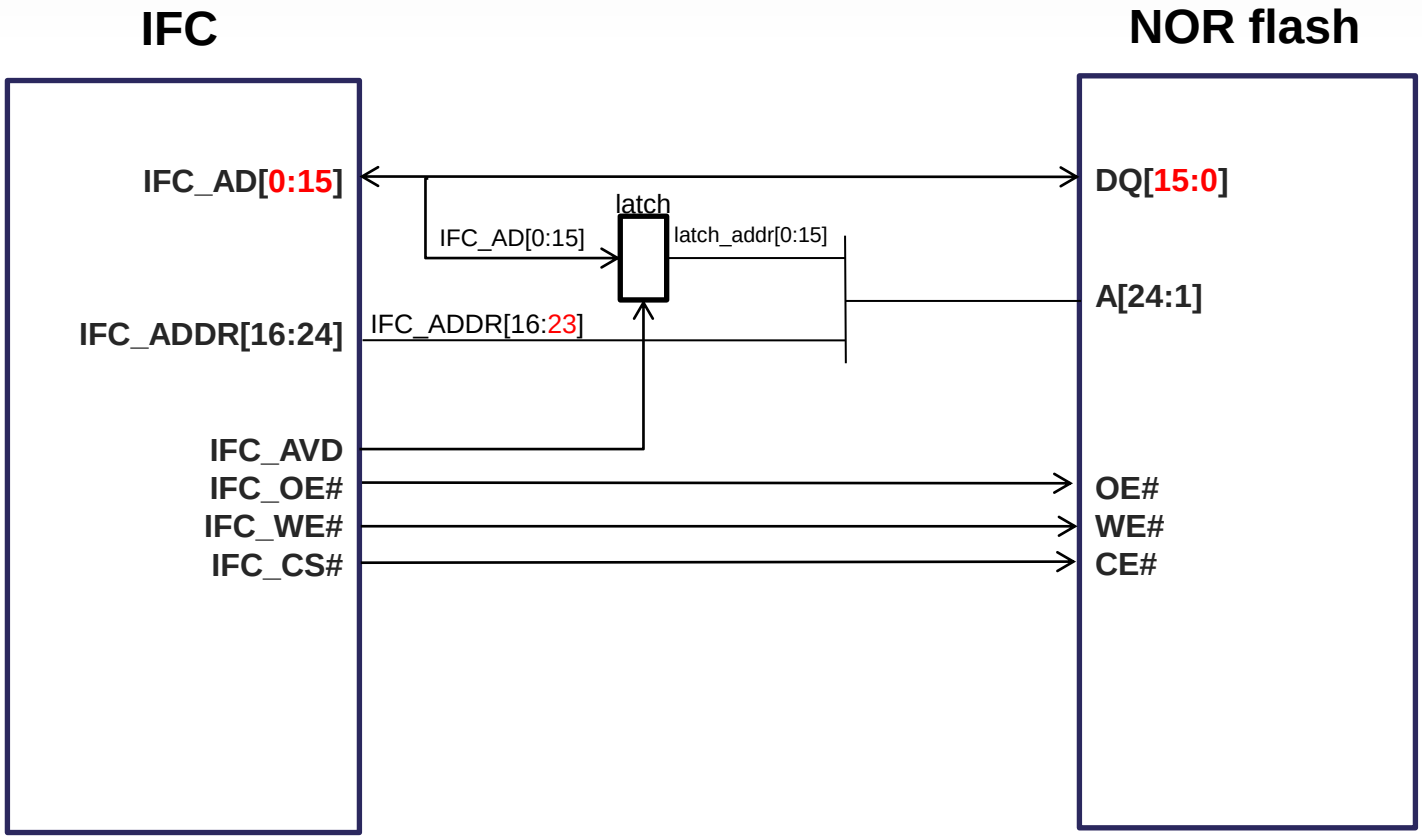
- Support standard asynchronous NOR flash
  - Same as GPCM
- Support page mode NOR flash
  - This is the improvement from GPCM for higher bandwidth
- No support for synchronous burst mode NOR flash



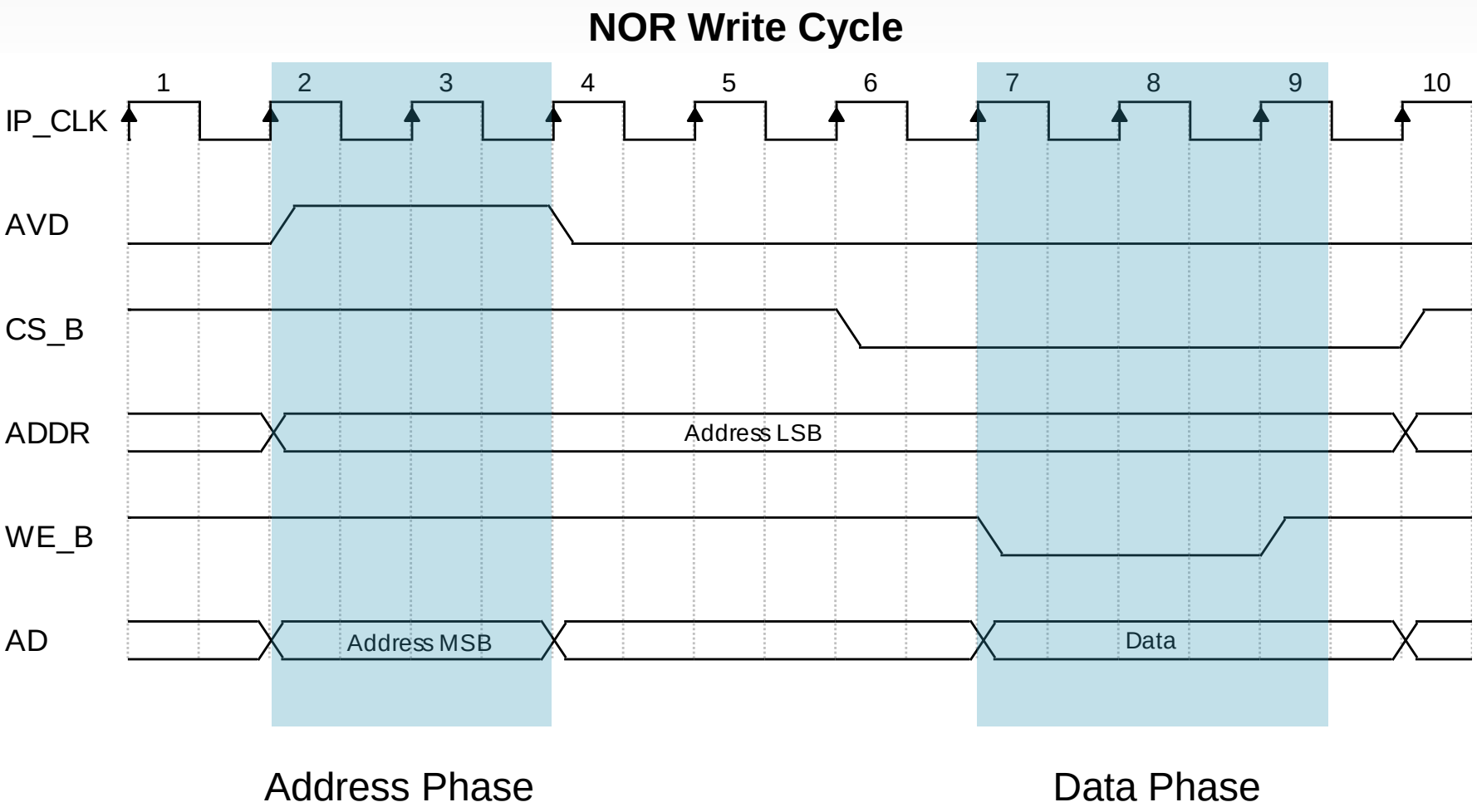
Interfacing IFC with regular 8-bit NOR flash



Interfacing IFC with regular 16-bit NOR flash

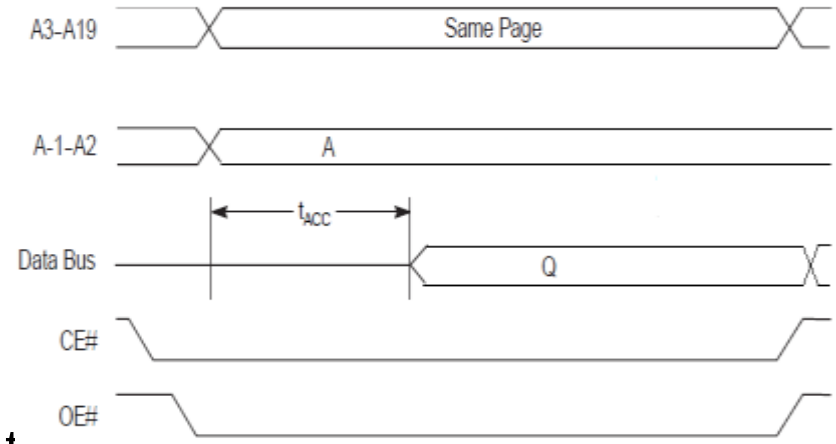


Address and Data Phase

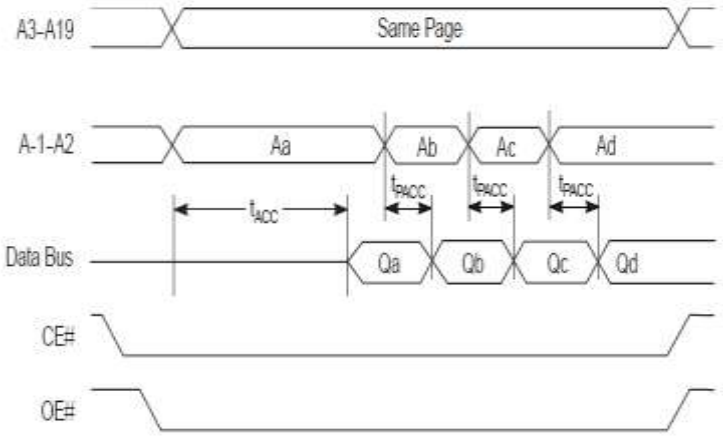


Page Mode Flash

Standard NOR flash **vs.** **Page Mode NOR**



t



For 16-bit interface and a 32-byte cache line read

Standard flash: 100ns x 16 = 1600 ns

Page mode: 100ns + 15 x 25ns= 475ns

Key points in the NOR setup

- IFC_ADDR[24] must be left out for the 16-bit port.

- When choosing the page-mode flash, the page size must be the same as the cache line, i.e., 32 bytes.

- After power on reset, CSORn[PGRD_EN] is set to 0.

  Software must set this bit to enable page read mode.

  (All page-mode flashes support standard mode access and they can be used for booting.)

- In order to take advantage of the page mode
  - Use DMA to copy the code from NOR flash to DDR
  - Enable cache

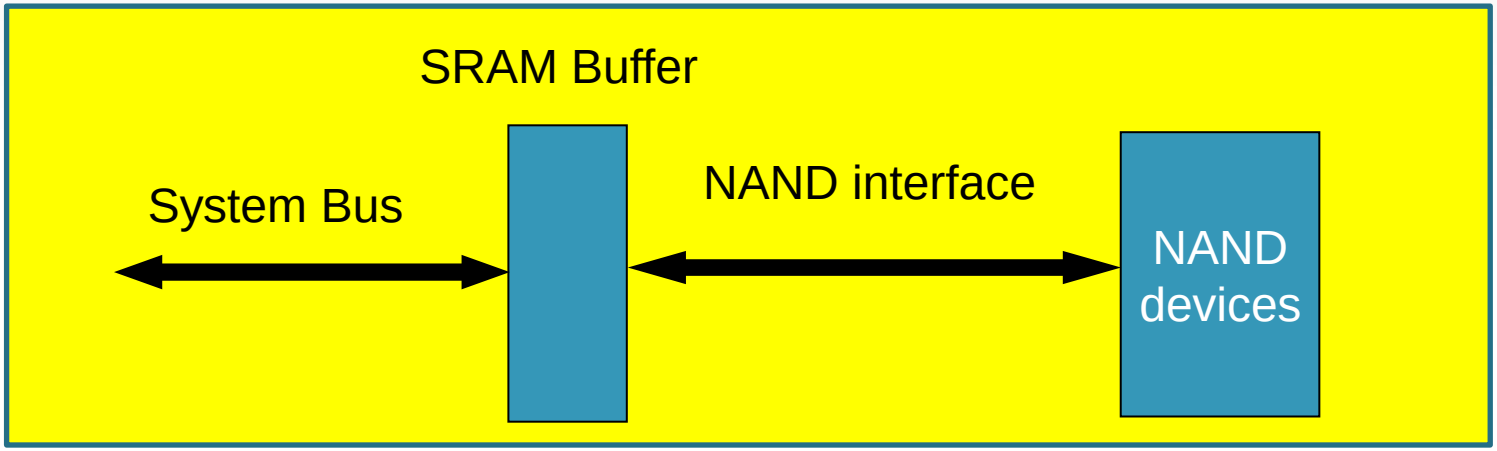
Overview of NAND Flash

- Higher-density / lower-cost than regular flash (NOR flash)
- IO device using commands to read/write
  - No address bus
  - Page-oriented, not suitable for random access
- Possible bit error
  - Usually a certain number of blocks are marked bad by manufacturer
  - During the operation, more blocks can go bad
  - ECC is a must
- No execute in place (XIP)
  - Long wait time for random address
  - Possible bit error only known after whole page read and ECC check

NAND Controller Features & Facts

- Support x8/x16 NAND devices
- ONFi (Open NAND Flash Interface) asynchronous interface with mandatory commands
- BCH coding for 4/8 bit error correction per block
  - IFC rev1.1 adds support of 24-bit/1K, 40-bit/1K
- Page size: Support 512 bytes, 2K and 4K
  - IFC rev1.1 adds support of 8K page
- Advance NAND commands like cache, copy-back and multi-plane programming
- Configurable block size, from 32 to 2048 pages per block

SRAM Buffer



- Due to the slowness of NAND devices, SRAM buffer is used to decouple the system bus from NAND accessing.
- System bus read/write to the memory bank(defined by CSPR[BA]/AMASK[AM]) actually accesses SRAM.
- For NAND write:

  (1) Data is transferred to SRAM; IFC is idle and free for other transactions.

  (2) Start NAND write with NANDSEQ_STRT; data transferred from SRAM to NAND

  During the write, SRAM must not be accessed
- For NAND read:

  (1) Start the NAND read with NANDSEQ_STRT; Data transferred from NAND to SRAM

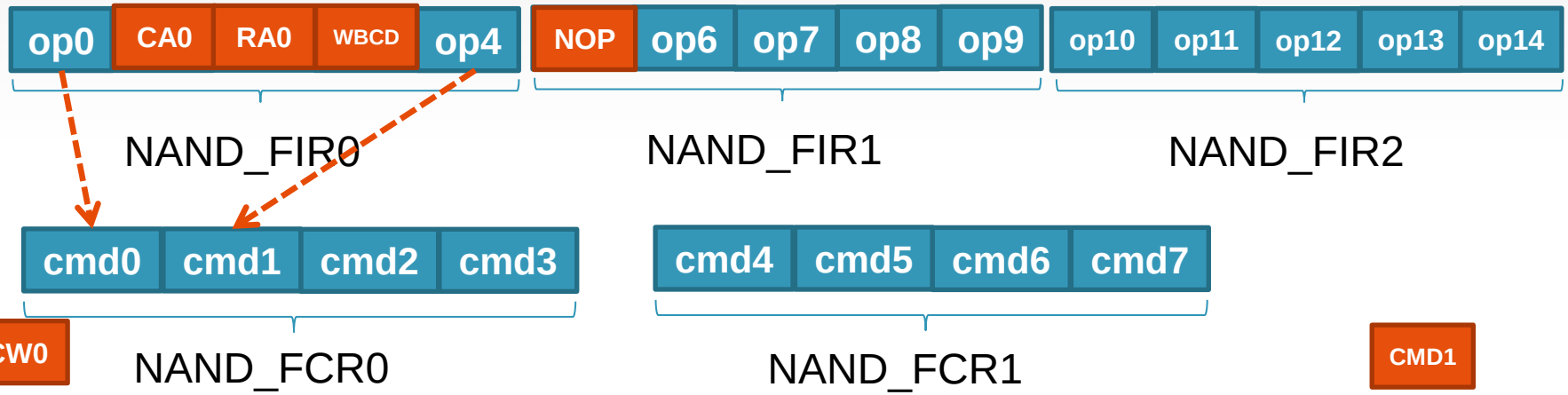
  SRAM must not be accessed during the read operation

  (2) Core accesses data from SRAM

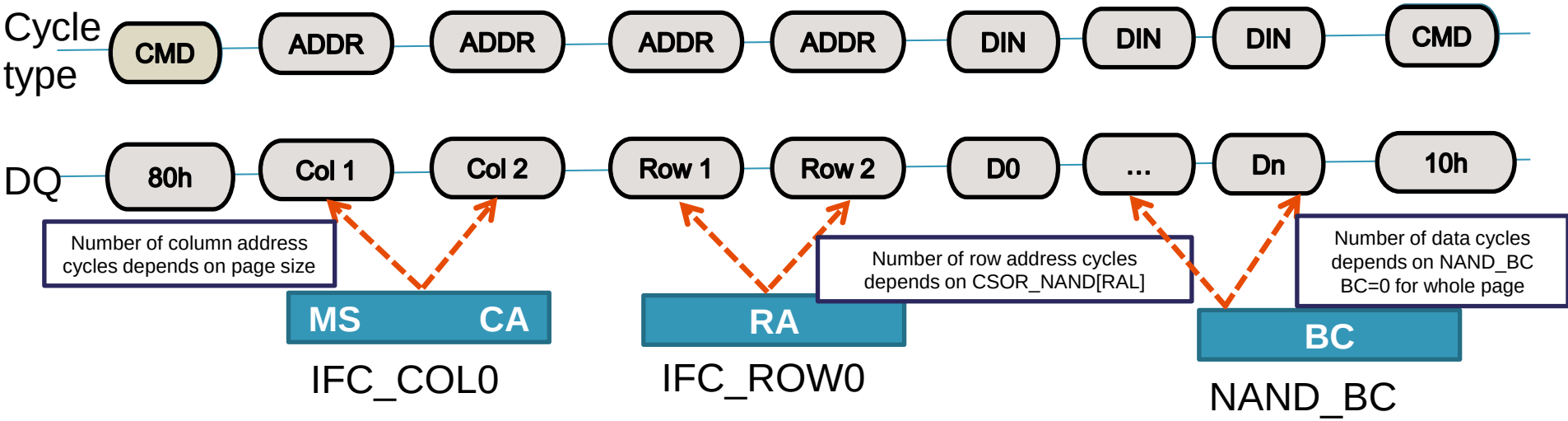
How Does NAND Controller Work

- Different vendors might have slightly different sequences
- Commands might be different, especially for small page NAND
- FCM takes a generic approach: User has flexibility/responsibility to define command sequence based on NAND
- NAND controller also supports a few standard commands
- Programming model
  1. Generate the command sequence
  2. AC timing control

How to generate the command sequence
Method1: NAND_FIR0-2



e.g. ONFi 2.2 Page Program



NAND_CSEL

CSEL

Select which CS

NANDSEQ_STRT

FIR_STRT

Start the operation

How to generate the command sequence Method 2: Use predefined sequences IFC_NANDSEQ_STRT

- **Setting to 1 triggers the operation. It is self-cleared**

Offset 0x0_1164 Access: Read/Write

	0	1 7	8	9 10	11	12 13	14	15 16	17	18 19	20	21 31
R W	NAND_FIR _STRT	—	AUTO_ ERS	—	AUTO _PGM	—	AUTO _CPB	—	AUTO_ RD	—	AUTO_ STAT_RD	—
Reset	All zeros											

Figure 12-39. NANDSEQ_STRT Register

- **Method 2:**

  FIR0-2 sequence defined automatically. Users still need to define other registers.

  AUTO_ERS: Erase

  AUTO_PGM: Program

  AUTO_CPB: Copy back

  AUTO_RD: Read

  AUTO_STAT_RD: Status read

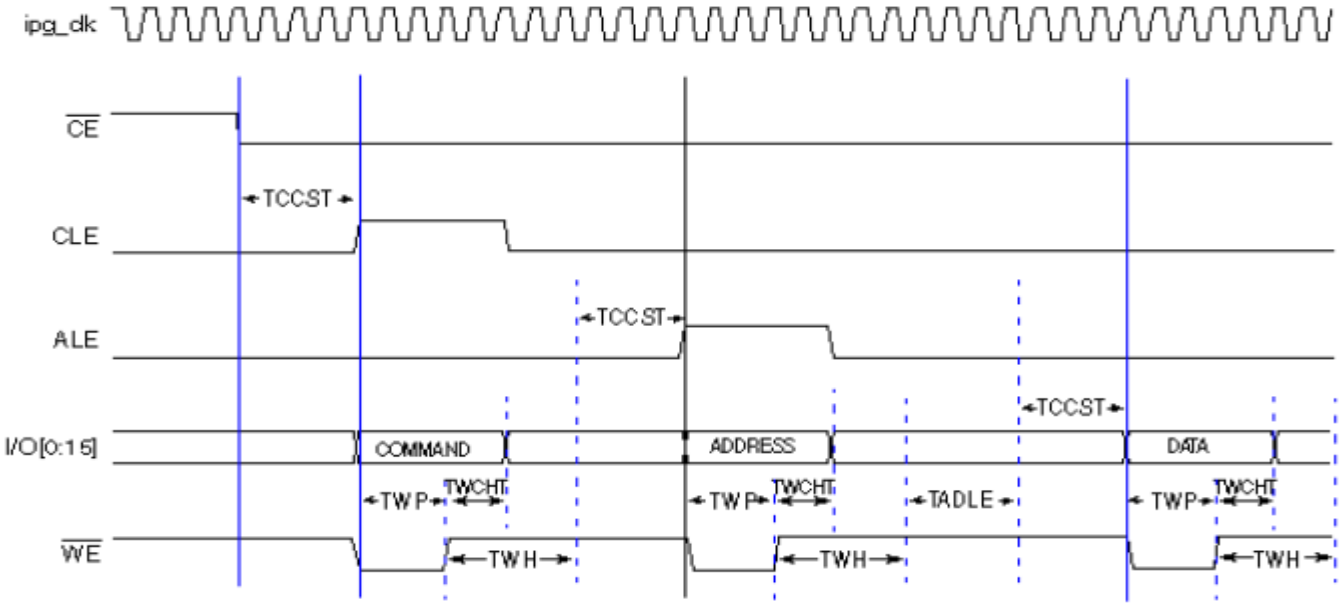
NAND AC Timing Control

- NAND interface is asynchronous – all the control and data signals have to maintain proper AC timing

- AC timing is defined in four registers

  IFC_FTIM0_CSn_NAND, IFC_FTIM1_CSn_NAND

  IFC_FTIM2_CSn_NAND, IFC_FTIM3_CSn_NAND



AC Timing Control:
Mapping Between ONFi Spec and IFC Registers

ONFi AC Timing Parameter	IFC control
tADL	FTIM1_CSx[TADLE]
tALS, tCLS, tDS, tWP	FTIM0_CSx[TWP] IFC drives ALE/CLE/DQx at the falling edge of WE#. TWP > max{tALS, tCLS, tDS, tWP}
tALH, tCLH, tDH,	FTIM0_CSx[TWCHT] IFC negates ALE/CLE/DQx at the same time after WE# high. TWCHT > max{tALH, tCLH, tDH}
tWH	FTIM0_CSx[TWH]
tRP	FTIM1_CSx[TRP]
tREH	FTIM2_CSx[TREH]
tRR	FTIM1_CSx[TRR]
tWB	FTIM1_CSx[TWBE]
tWC	TWH+TWP > tWC

AC Timing Control

ONFi AC Timing Parameter	IFC control
tRC	TRP + TREH > tRC
tWW	FTIM3_CSx[TWW]
tCS	
tCH	Command latch cycle: Satisfied automatically during Data write cycle: FTIM0_CSx[TWH] > tCH
tRHZ	CSORx_NAND[TRHZ] > tRHZ
tREA	FTIM2_CSx[TRAD] Data sampling point tRP-tREA+tRHOH
tWHR	FTIM2_CSx[TWHRE]
tIR	Satisfied automatically since it is much smaller than tWHR
tRHW	CSORx_NAND[TRHZ] > tRHW

Booting from NAND & Reset Configuration

- NAND controller is selected as the boot ROM location.

  cfg_rom_loc[0:3]:

1000: 8-bit, 512B page	1100: 16-bit, 512B page
1001: 8-bit, 2KB page	1101: 16-bit, 2KB page
1010: 8-bit, 4KB page	1110: 16-bit, 4KB page

- NAND controller searches for a valid boot block

  cfg_ifc_pb[0:2]: 32-2K pages per block

  cfg_ifc_flash_mode: Bad block indication location setting

- Reads data until 8K SRAM buffer is filled.

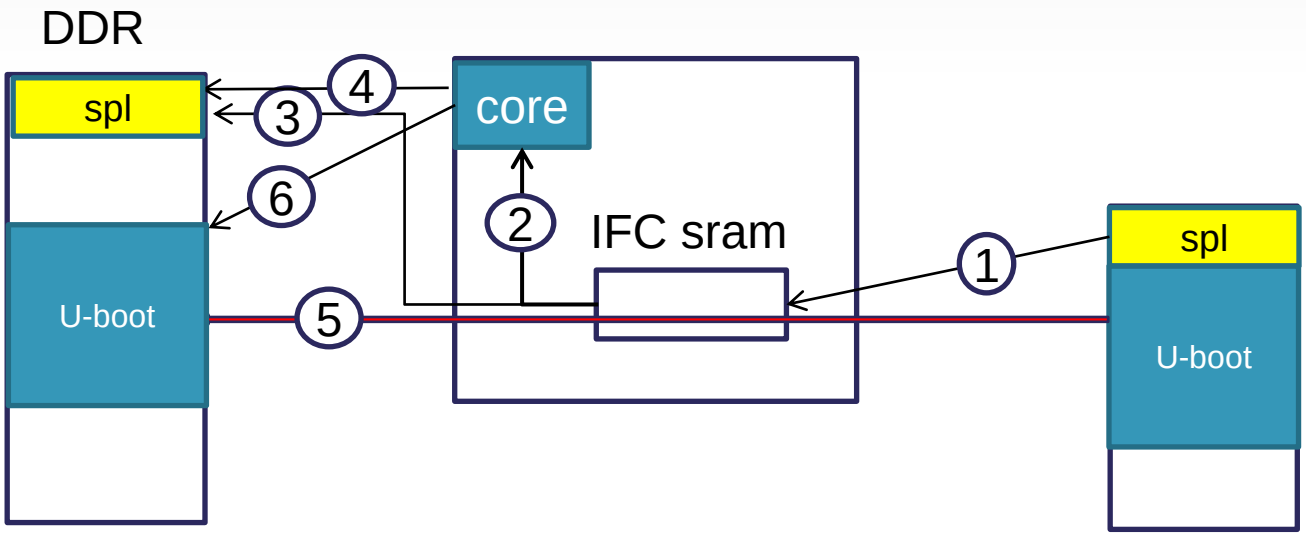
  cfg_ifc_ecc[0:1]: ECC disable; 4b; 8b

  ECC errors are corrected if possible and enabled.

  Boot stops if error is uncorrectable, /HRESET_REQ is asserted.

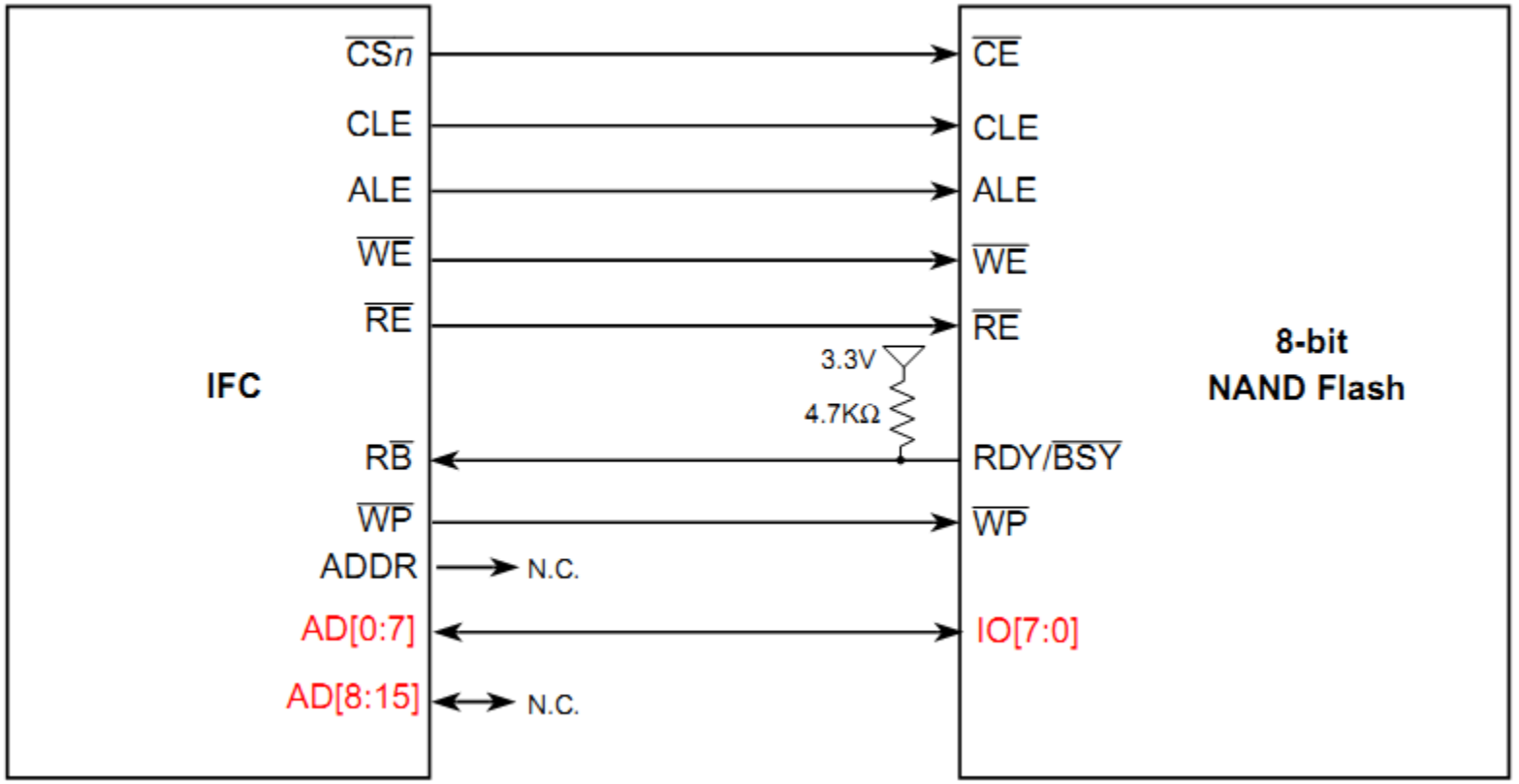
- The CPU is released to start fetching instructions from the SRAM buffer.

U-boot Booting process



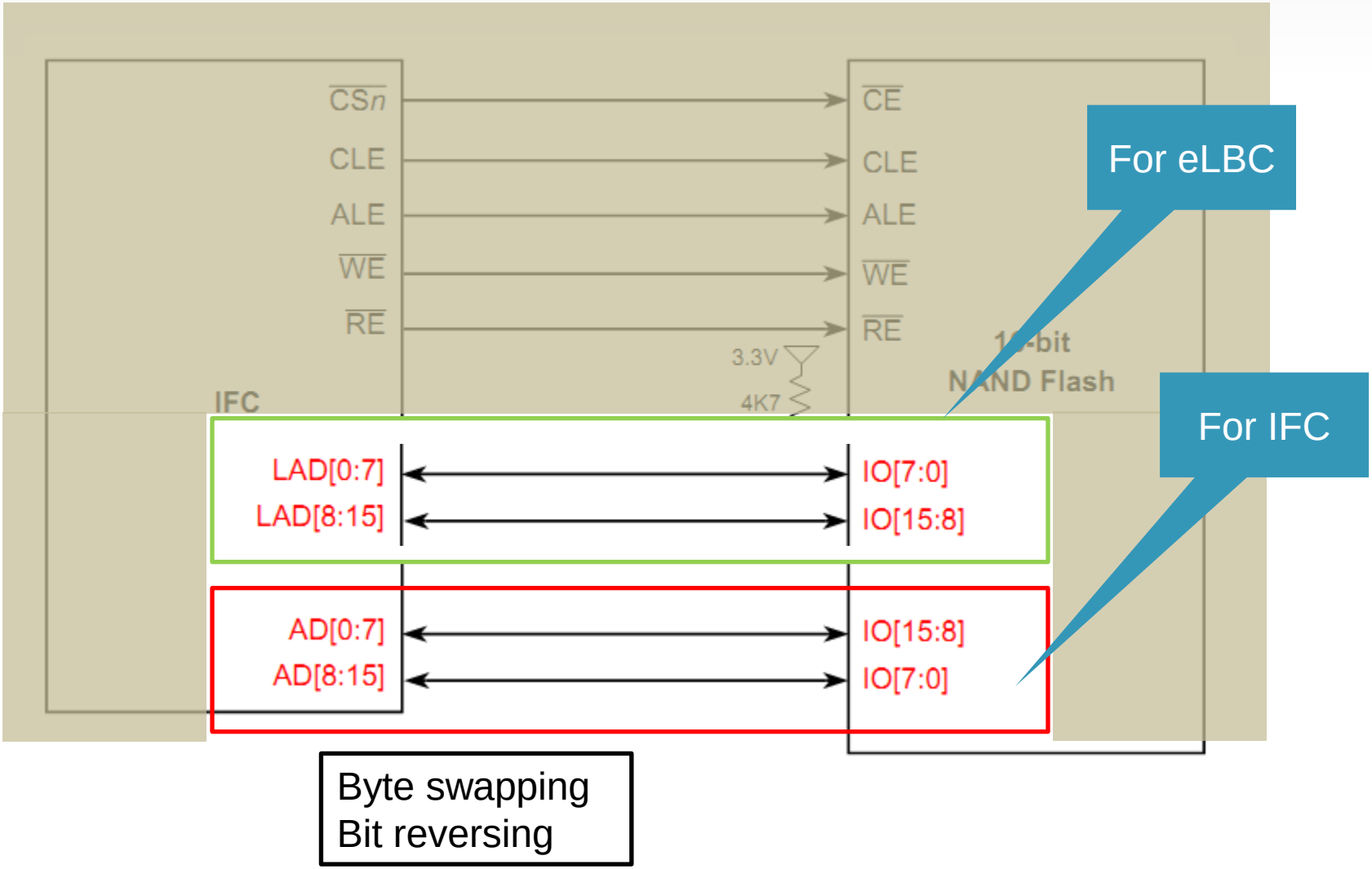
- Step 1: NAND controller loads automatically preloader spl to IFC SRAM
- Step 2: Core executes code out of SRAM
- Step 3: spl relocates itself to DDR to empty sram for NAND controller
- Step 4: spl continues executes from DDR
- Step 5: spl copies u-boot from NAND to DDR
- Step 6: core jumps to DDR to start u-boot

Interfacing 8-bit NAND with P1010



Bit reversing

Interfacing 16-bit NAND with P1010



GPCM Controller

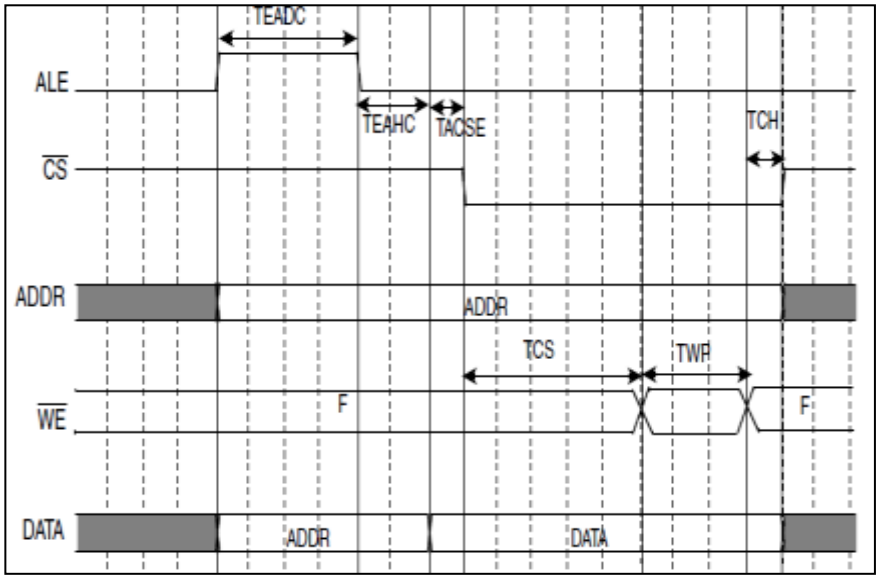
- GPCM supports 2 modes:
  - Normal GPCM
  - Generic ASIC
- Enable GPCM and select mode
  - CSPRn[MSEL]: 0b10 for GPCM
  - CSORn[GPMODE]: 0 for normal GPCM, 1 for generic ASIC
- Normal GPCM is similar to eLBC GPCM with new programming model
- Generic ASIC is a new function

Normal GPCM Mode: Write

- Normal GPCM can be used to generate standard NOR flash interface compatible control signals

- ALE timing controlled by **FTIM0_CSn**
  - **TACSE**: Address to CS assertion
  - **TEADC**: Pulse width of ALE
  - **TEAHC**: ALE to address hold time

- WE timing for read controlled by **FTIM2_CSn**
  - **TCS**: CS to WE assertion time
  - **TWP**: WE pulse width
  - **TCH**: WE negation to CS negation



GPCM Write

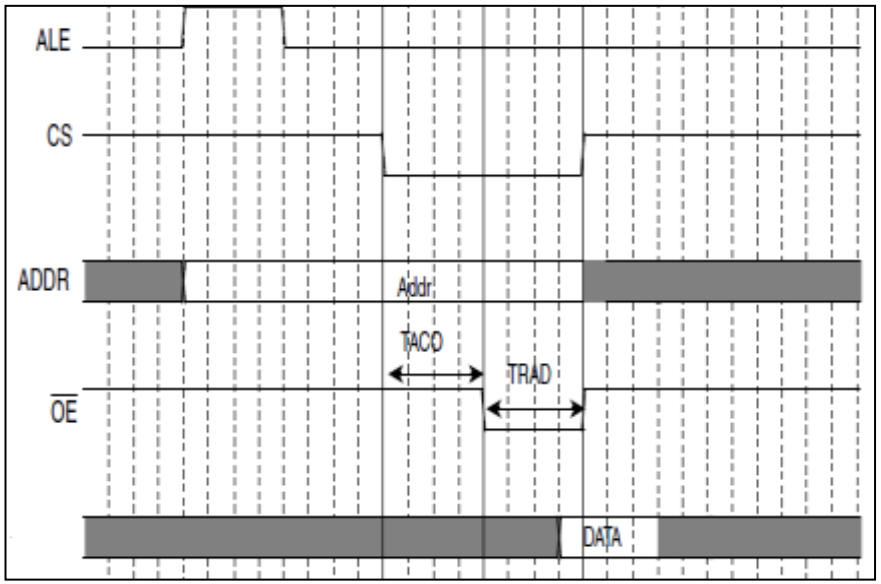
Normal GPCM Mode: Read

ALE timing controlled by **FTIM0_CSn**

- **TACSE**: Address to CS assertion
- **TEADC**: Pulse width of ALE
- **TEAHC**: ALE to address hold time

OE timing for read controlled by **FTIM1_CSn**

- **TACO**: CS to OE time
- **TRAD**: OE pulse width



GPCM Read

Normal GPCM Mode: External Termination

- Option for external termination by IFCTA

  Read - CSORn[RGETA]

  Write - CSORn[WGETA]

  0: Terminated by internal TRAD counter for read or TWP for write or IFCTA if it is asserted earlier than internal timer expiration

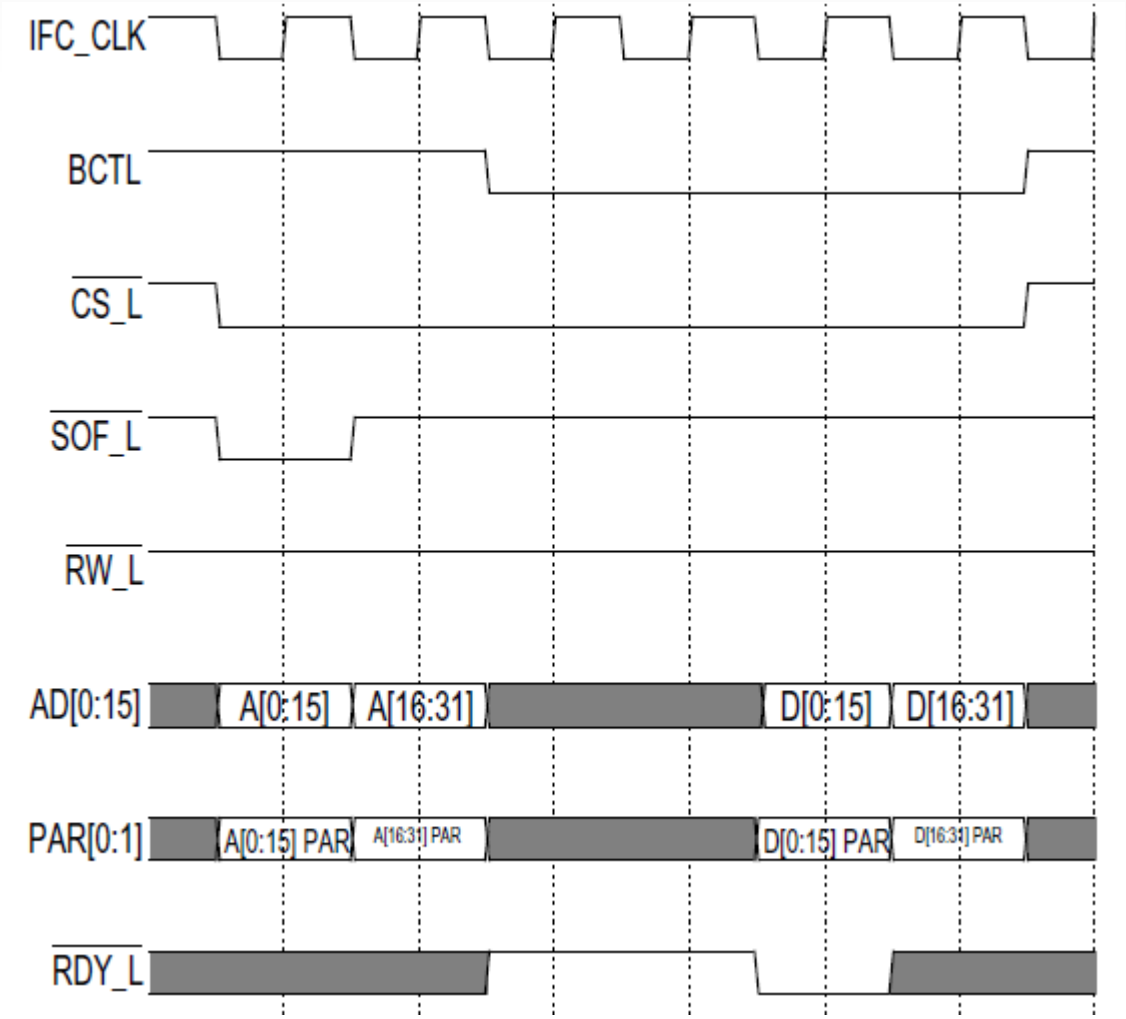
  1: Only terminated by assertion of IFCTA, not internal timer

- This feature is useful if response time is variable

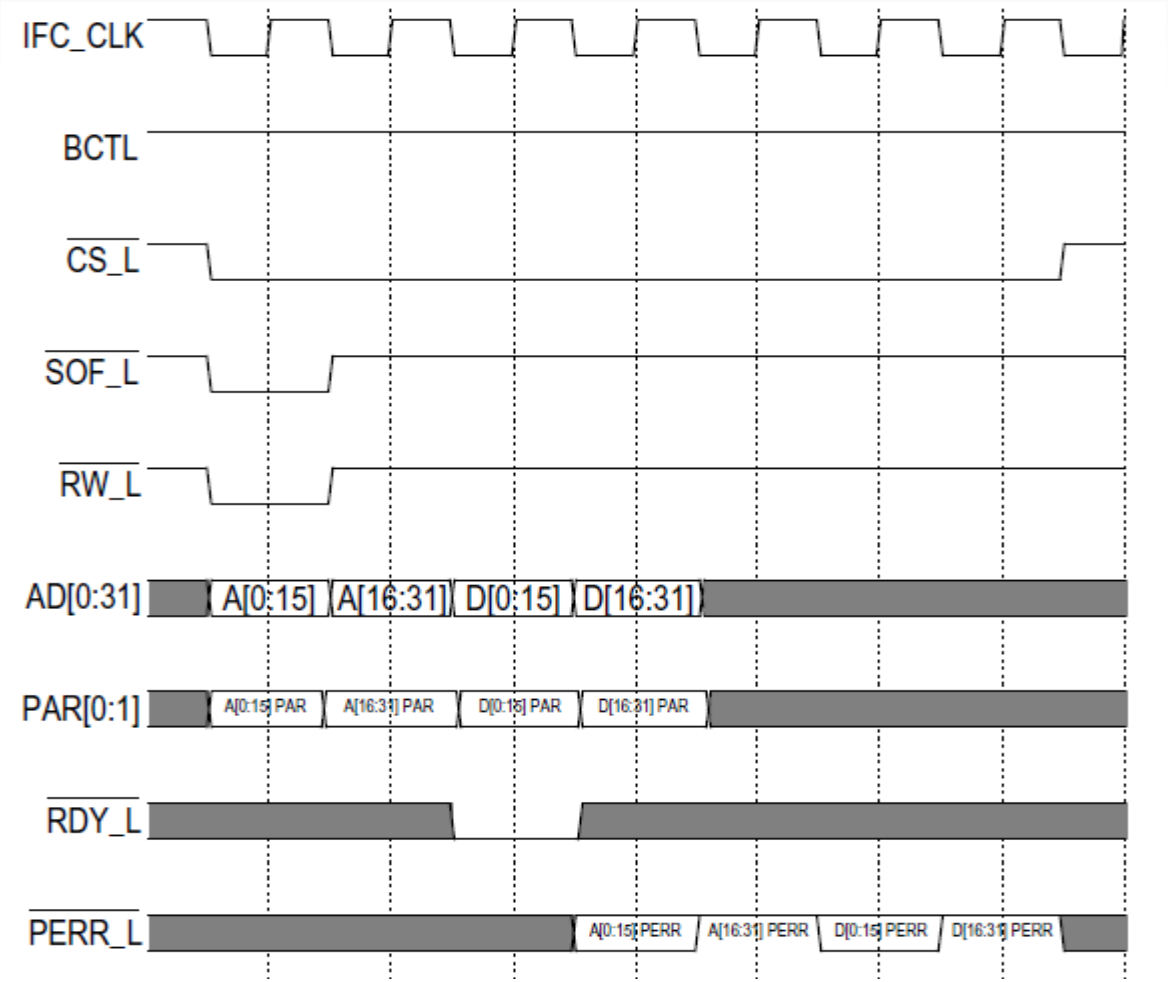
Generic ASIC

- A simple interface useful for talking to FPGA
- IFC supports the following features on GASIC interface:
  - Support for x8/16-bit device
  - Address and data are shared on AD I/O bus; dedicated address pins are not used
  - Following address and data sequences will be supported on I/O bus
    - 16-bit I/O: AADD
    - 8-bit I/O: AAAADDDD
  - Configurable even/odd parity on address/data bus supported
  - Parity error detection supported
- GASIC interface does *not* support:
  - Boot from GASIC
  - Burst transaction

Generic ASIC: 16-bit Interface Read



Generic ASIC: 16-bit Interface Write



Q&A

- How may I help you?

Freescale on Kaixin
Tag yourself in photos
and upload your own!

Weibo?
Please use hashtag
#FTF2012#

Session materials will be posted @ www.freescale.com/FTF

freescale™


freescale™