Important: **FA626TE55EE0001HD0AG (2.1.0) is currently retained by Faraday. Please contact Faraday local service team for further information.**

FA626TE | FA626TE55EE0001HC0HA
FA626TE54EE0001HC0HA | FA626TE55EE0001HD0AG

32-BIT RISC PROCESSOR

Data Sheet
Rev.: 2.17
Issue Date: January 2011

FARADAY

REVISION HISTORY

FA626TE|FA626TE55EE0001HC0HA|FA626TE54EE0001HC0HA|FA626TE55EE0001HD0AG Data Sheet

Date	Rev.	From	To
Jul. 2007	1.0	-	Original
Oct. 2007	1.1	-	Added a new IP: FA626TE55EE0001HD0AG
Dec. 2007	1.2	-	Updated the IP version
Feb. 2008	1.3	-	Updated the content due to ECO
Feb. 2008	1.4	-	Updated the IP version
Mar. 2008	1.5	-	Updated the IP version
Apr. 2008	1.6	-	Added a new IP: FA626TE54EE0001HC0HA
May 2008	1.7	-	• Modified the power-down control in Section 9.3 • Modified the contents in Table 2-8
May 2008	1.8	-	Added a new IP: FA626TE
Jun. 2008	1.9	-	• Modified the cp15 registers 1-0 and 1-1 • Added Bit 14 and Bit 16 on page 14
Jun. 2008	2.0		Updated the IP version of FA626TE55EE0001HC0HA from (1.0.0) to (1.1.0)
Aug. 2008	2.1	-	Updated the IP version of FA626TE55EE0001HD0AG from (0.2.0) to (0.3.0)
Aug. 2008	2.2	-	Updated the IP version of FA626TE55EE0001HD0AG from (0.3.0) to (1.0.0)
Oct. 2008	2.3	-	Modified the scratch pad size in Section 1.7 and Section 8.1
Jan. 2009	2.4	-	Updated the IP versions: • FA626TE55EE0001HC0HA from (1.1.0) to (2.0.0) • FA626TE from (0.0.1) to (0.1.0)
Aug. 2009	2.5	-	Updated the IP versions: • FA626TE55EE0001HD0AG from (1.0.0) to (1.1.0) • FA626TE from (0.1.0) to (1.2.0)
Sept. 2009	2.6	-	Revised the pin descriptions
Dec. 2009	2.7	-	Updated the IP version of FA626TE to (1.3.0)
Jan. 2010	2.8	-	Updated the IP version of FA626TE55EE0001HD0AG to (1.1.1)
Feb. 2010	2.9	-	Added test pin signal descriptions in Table 11-7 and Table 11-9

Date	Rev.	From	To
Mar. 2010	2.10	-	Updated the IP version of FA626TE55EE0001HD0AG to (1.2.0)
May 2010	2.11	-	Updated the descriptions of the Scratchpad Interface Signals in Table 11-5
Jun. 2010	2.12	-	Updated the IP version of FA626TE55EE0001HD0AG to (2.0.0)
Jul. 2010	2.13	-	Updated the IP version of FA626TE to (1.3.1)
Aug. 2010	2.14	-	Updated the IP version of FA626TE55EE0001HC0HA from (2.0.0) to (2.1.0)
Aug. 2010	2.15	-	Updated the IP version of FA626TE55EE0001HD0AG to (2.1.0)
Nov. 2010	2.16	-	Updated the IP version of FA626TE to (1.3.2)
Jan. 2011	2.17	-	Added the AXI DMA specifications

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

This chapter contains the following sections:

FA626TE is an ultra high-speed, general-purpose, 32-bit embedded RISC processor. It includes a CPU core, separate instruction/data caches, separate instruction/data scratchpads, a write buffer, a Memory Management Unit (MMU), and a JTAG ICE interface. The CPU core uses the Harvard architecture design with eight pipeline stages. To improve the overall performance, each of the FA626TE CPU cores also contains a Branch Target Buffer (BTB) and Return Stack (RS) to reduce the branch penalty.

The FA626TE CPU cores are instruction-compatible with the ARM V5TE® architecture. This CPU core uses four AHB or AXI interfaces with a configurable width of 32-bit or 64-bit to communicate with the external memory and devices. The FA626TE CPU cores are suitable for a wide range of applications, especially for those requiring high performance and low power consumption. The FA626TE CPU core is fully synthesizable and its single-phase clock-based architecture makes the SoC integration a very easy task.

The block diagram of the FA626TE CPU cores is shown in Figure 1-1. The detailed description of each functional block can be found in the following sections.

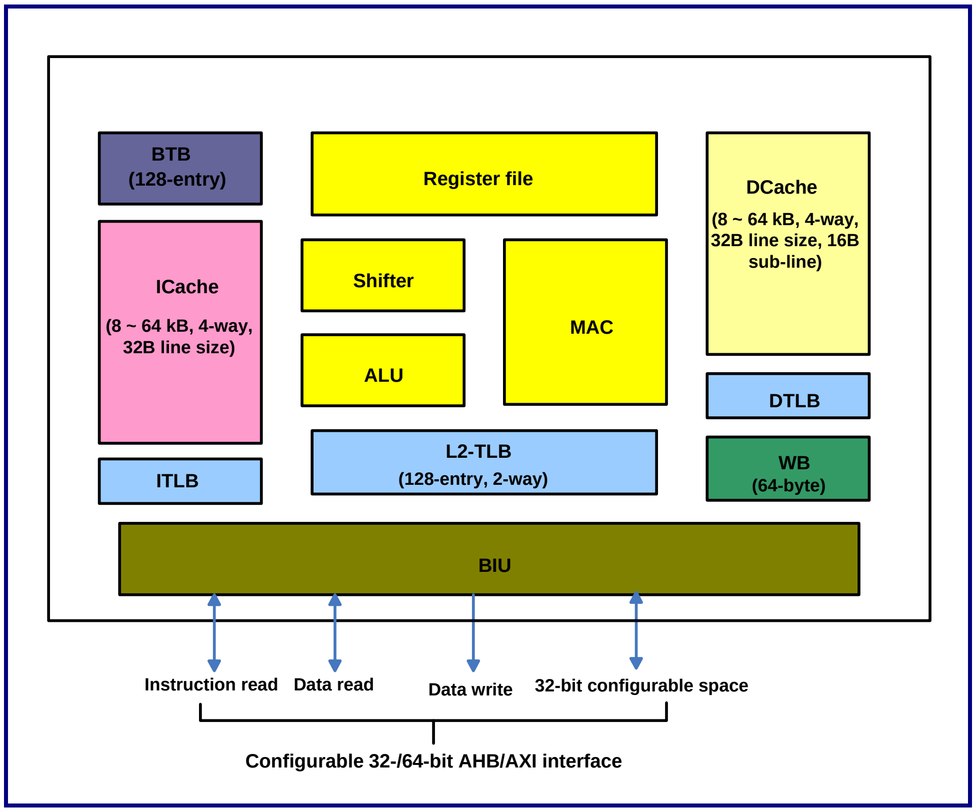
BTB
(128-entry)
Register file
DCache
(8 ~ 64 kB, 4-way, 32B line size, 16B sub-line)
ICache
(8 ~ 64 kB, 4-way, 32B line size)
Shifter
MAC
ALU
DTLB
L2-TLB
(128-entry, 2-way)
ITLB
WB
(64-byte)
BIU
Instruction read
Data read
Data write
32-bit configurable space
Configurable 32-/64-bit AHB/AXI interface

Figure 1-1. Functional Block Diagram

1.1 Version of the IP

IP Name	Process	IP Version
FA626TE55EE0001HC0HA	UMC 0.13 µm logic HS FSG process	2.1.0
FA626TE54EE0001HC0HA	UMC 0.13 µm logic HS FSG process	0.2.0
FA626TE55EE0001HD0AG	UMC 90 nm logic SP/RVT Low-K process	2.1.0
FA626TE	FA626TE 32-bit RISC processor	1.3.2

1.2 CPU Core

The FA626TE cores are fully compliant with the ARM V5TE® architecture, including the V5TE instruction set. The ARM V5TE® instruction set is well-documented in the ARM Architecture Reference Manual, second edition. The CPU cores implements the Harvard architecture design with eight-stage pipelines, which consist of Fetch, Instruction, Decode, Register Access, Shift, Execution, Memory, and Write stages. There are a total of 30 general-purpose registers and six processor status registers. Each of the FA626TE cores provides seven processor modes, including the Supervisor, System, FIQ, IRQ, Abort, Undefined, and User modes.

1.3 Branch Prediction Unit (BPU)

The Branch Prediction Unit (BPU) can improve the processor performance through three branch prediction mechanisms: Branch Target Buffer (BTB), Return Stack (RS), and Static Branch Prediction (SBP). With an accurate branch prediction, BTB can resolve most of the control dependency and reduce the branch penalty. The FA626TE BTB is a 128-entry direct-map structure with the 2-bit counter algorithm in the branch prediction, and provides all invalidate BTB-entry operations.

The Return Stack (RS) is a two-entry buffer that stores the returned address of a procedure call. The returned address is pushed into RS when a procedure call occurs. When the returned instruction is predicted, the returned address is popped from RS. An empty RS gives no prediction.

The Static Branch Prediction (SBP) is used when a branch misses BTB. SBP always performs prediction when the unconditional branches (B/BL) are taken. For conditional branches, the backward branches are predicted as taken and the forward branches are predicted as not taken.

1.4 Instruction Cache (ICache)

The Instruction Cache (ICache) applies the locality of a program to improve the performance of a processor. The most recently used instructions are stored in the faster local memory, ICache. As a result, when the instructions are executed next time, the processor can access the instructions from ICache, instead of taking a long access latency to fetch the instructions from a slower external memory. ICache has a 4-way set-associate architecture and can be configured as 4K, 8K, 16K, 32K, or 64K in size. A simplified Least Recently Used (LRU) algorithm is used as the replacement strategy. The FA626TE cores provide several management instructions for ICache invalidation, pre-fetch, and locking.

1.5 Data Cache (DCache)

The Data Cache (DCache) applies the locality of a program to improve the performance of a processor. The most recently used data are stored in a faster local memory, DCache. As a result, when the data are needed next time, the processor can access them from DCache instead of taking a long latency to access the data from a slower external memory. DCache has a 4-way set-associate architecture and can be configured as 4K, 8K, 16K, 32K, or 64K in size. A simplified LRU algorithm is implemented as the replacement strategy. The FA626TE cores provide several management instructions for DCache invalidation, clean, pre-fetch, and locking.

1.6 Memory Management Unit (MMU)

The Memory Management Unit (MMU) provides the address translation and permission check mechanism for memory access. The FA626TE MMU implements the two-level TLB structure. Level-1 TLB includes an ITLB for the instruction access look-up and a DTLB for the data access look-up. Both look-ups are 8-entry fully-associate. Level-2 TLB (UTLB) is a unified 2-way set-associate TLB structure. TLBs cache the most recently used page descriptors for the address translation, which greatly improves the overall performance. Once UTLB misses, the page table walk will be completed automatically by the hardware.

The FA626TE MMU is compatible with the MMU defined in the ARM Architecture Reference Manual, second edition, with some minor differences due to the inherent differences in cache and TLB implementation.

1.7 IScratchpad and DScratchpad

A scratchpad is a fast on-chip SRAM that is located near the processor core. The performance-critical code or data can be pre-fetched to the scratchpads and be executed at full CPU speed. This is particularly useful for embedded applications. The size of a scratchpad can be 1 kB, 2 kB, 4 kB, 8 kB, 16 kB, 32 kB, 64 kB, or 128 kB. The FA626TE CPU will automatically fill the entire IScratchpad when the first scratchpad instruction is accessed. DScratchpad needs to be filled by software. The sizes of IScratchpad and DScratchpad can be programmed as 1 kB, 2 kB, 4 kB, 8 kB, 16 kB, 32 kB, 64 kB, or 128 kB. The base addresses of IScratchpad and DScratchpad can be programmed at the CR9 register by using the physical address. Both IScratchpad and DScratchpad of FA626TE are removable.

1.8 Bus Interface Unit (BIU)

The Bus Interface Unit (BIU) accepts the requests for memory accesses from the CPU cores and executes instructions and data accesses through the external system bus. Four bus ports are designed to provide a high-bandwidth interface. The Instruction Read (IR) port serves the instruction reading. The Data Read (DR), Data Write (DW), and Peripheral Ports serve as the data access. Each of the four interfaces is an AHB or AXI interface.

1.9 Write Buffer (WB)

The Write Buffer (WB) can hide the write latency to the next level of memory hierarchy and reduce the write traffic. Without the write buffer, the CPU cores must be stalled when writing to a slower external memory. The write buffer accommodates the slower memory writes so that the CPU cores do not have to wait until the write operation is complete. This improves the overall performance. The FA626TE write buffer includes an address buffer and a data buffer. The size of the data buffer is eight double-words, while the size of the address buffer is eight words.

1.10 ICE

The FA626TE core provides an ARM-compatible JTAG-style ICE interface. The ICE module serves as the program for debugging FA626TE through this interface. The standard ARM debuggers, such as AXD™ or RealView™, can be used for debugging. For more details of the FA626TE ICE specifications, please refer to Chapter 10.

1.11 Power-saving Control Unit

The power-saving control unit controls the processor global clocking to reduce the operating power. It also provides the power-saving mode. When the software program detects that the CPU is in the idle state for a given amount of time, it can force the processor to enter the power-saving mode. In the power-saving mode, the processor stops the instruction execution and enters an idle state. The system logic can then stop the whole processor clock to save power. While in the power-saving mode, the processor can be awoken through either an interrupt or an ICE activity. The start-up time is 16 cycles if the processor is awoken from an idle mode.

1.12 Coprocessor Transfer Instruction

FA626TE only supports the MCR and MRC instructions for CP15 (MMU), CP14 (ICE), and CP8 (Power-on trap). All other coprocessor-related instructions, such as LDC, SDC, CDP, LDC2, STC2, MCR2, MRC2, MCRR, MRRC, or a coprocessor other than CP8, CP14, or CP15, will cause undefined exceptions.

Chapter 2

Programming Model

This chapter contains the following sections:

2.1 General Description

FA626TE implements the full ARM V5TE® instruction set. “T” and “E” refer to the thumb mode and the DSP-extension instructions, respectively. Please refer to the ARM Technical Reference Manual, second edition, for detailed descriptions on the instruction set.

For the data abort model, FA626TE implements a data abort base restore model. That is, when a data abort exception occurs during the execution of a memory access instruction, the base address will be restored to the value contained in the register before the memory access instruction is executed. This model dramatically reduces software efforts for the data abort exception handler.

FA626TE provides three coprocessors: CP15, CP14, and CP8. CP15 is a memory management coprocessor that provides the translation function from the virtual address to the physical address and the facility to control the caches, TLB, scratchpads, write buffer, and some other FA626TE behaviors. CP14 is a debugging control coprocessor. CP8 is a coprocessor that controls the clock and power-down status. Each coprocessor defines a set of registers. The register definition of CP15 is described in detail in the next section. Please refer to Chapters 8 and 9 for details on CP8 and CP14. All these registers are accessed through the MCR or MRC instruction.

2.2 CP15 Register Description

The memory management coprocessor (CP15) provides a set of control registers (CRs) that control the caches, TLB, scratchpad, write buffer, and some other behaviors of FA626TE. The CP15 register map is shown in Table 2-1. Programmers can access these registers through the MCR or MRC instruction. All CP15 registers can only be written in the non-user mode. Writing these registers in the user mode may cause an “undefined instruction” exception. Register 7 and register 8 define the system instructions as shown in Table 2-4 and Table 2-7.

Table 2-1. CP15 Register Map

Control Register #	Symbol	Access	Description
0-0	ID code	Read only	ID code register
0-1	CTR	Read only	Cache type register
0-3	TTR	Read only	TLB type register
1-0	CFG	R/W	Configuration register

Control Register #	Symbol	Access	Description
1-1	AUX	R/W	Auxiliary control register
2	TTB	R/W	Translation table base register
3	DAC	R/W	Domain access control register
4	-	-	Reserved
5	FSR	R/W	Fault status register
6	FAR	R/W	Fault address register
7	COPR	Write only	Cache operation register
8	TOPR	Write only	TLB operation register
9-0	DCL/ICL	R/W	Data/Instruction cache lockdown
9-1	DSC/ISC	R/W	Data/Instruction scratchpad configuration register
10	TLBL	Write only	TLB lockdown register
11	-	-	Reserved
12	-	-	Reserved
13	PID	R/W	Process ID register
14	-	-	Reserved
15_2	PRGN	R/W	Peripheral port region definition register
15	TEST	-	Test register

2.2.1 CR0-0 Identification Code Register (ID)

Read only Power-on Default: 0x66056261

[31:24]	[23:16]	[15:4]	[3:0]
IMP	ARCH	PART	VER

This register contains information for the identification code of a 32-bit processor. Each field is defined in the following sections.

Bits[31:24]: Implementer

ARCH	Architecture
0x66	Faraday
Others	Reserved

Bits[23:16]: Architecture Version

ARCH	Architecture
0x05	ARM V5TE
Others	Reserved

Bits[15:4]: Chip Part Number

PAR	Chip
0x626	FA626TE
Others	Reserved

Bits[3:0]: Chip Version Number

VER	Chip
0x1	1st version
Others	Reserved

Programmers can access this ID code register via MRC instruction with the opcode_2 field set to any value other than one. The following instruction is an example to access this register:

MRC p15, 0, Rd, c0, c0, op2 ; Read the ID code register, op2 can be 0, 2, 4, 5, 6, or 7.

2.2.2 CR0-1 Cache Type Register (CTR)

Read only Power-on Default: 0x0F192192

[31:25]	24	[23:21]	[20:18]	[17:15]	14	[13:12]
0000111	1	000	DSIZE	DASS	0	DLEN

[11:9]	[8:6]	[5:3]	2	[1:0]
000	ISIZE	IASS	0	ILEN

This register contains information about the size and architecture of the instruction cache and the data cache. Each field is defined below:

Bits[28:21]: Reserved

Bits[20:18]: DSIZE. Data cache size. Default is set to 3'b110.

Size Field	Cache Size
3'b000	512 B
3'b001	1 kB
3'b010	2 kB
3'b011	4 kB
3'b100	8 kB
3'b101	16 kB
3'b110	32 kB
3'b111	64 kB

Bits[17:15]: DASS. Data cache associability. Default is set to 3'b010.

ASS Field	Associativity
3'b000	Direct mapped
3'b001	2-way
3'b010	4-way
3'b011	8-way
3'b100	16-way
3'b101	32-way
3'b110	64-way

ASS Field	Associativity
3'b111	128-way

FARADAY

Bits[13:12]: DLEN. Data cache line length. Default is set to 2'b10.

LEN Field	Cache Line Length
2'b00	2 words (8 bytes)
2'b01	4 words (16 bytes)
2'b10	8 words (32 bytes)
2'b11	16 words (64 bytes)

Bits[8:6]: ISIZE (Instruction cache size). Default is set to 3'b110.

The encoding is the same as DSIZE.

Bits[5:3]: IASS (Instruction cache associability). Default is set to 3'b010.

The encoding is the same as DASS.

Bits[1:0]: ILEN (Instruction cache line length). Default is set to 2'b10.

The encoding is the same as DLEN.

Programmers can access this Cache Type register through the MRC instruction with the opcode_2 field set to one. The following instruction is an example to access this register:

MRC p15, 0, Rd, c0, c0, 1   ; Read CTR

2.2.3 CR0-3 TLB Type Register (TTR)

Read only   Power-on Default: 0x0000004

[31:4]	[3:1]	0
Reserved	TASS	T

Bits[3:1]: TASS. TLB associability Default: 3'b001.

ASS Field	Associativity
3'b000	Reserved
3'b001	2-way
3'b010	4-way
3'b011	8-way

ASS Field	Associativity
3'b100	16-way
3'b101	32-way
3'b110	64-way
3'b111	128-way

Bit 0: T. TLB type. Zero indicates unified TLB, and one indicates separated TLB. Default is set to 1'b0.

Programmers can access this TLB Type register through the MRC instruction with the opcode_2 field set to any value other than one. The following instruction is an example to access this register:

MRC p15, 0, Rd, c0, c0, op2; Read the TLB type register, op2 can be 0, 2, 4, 5, 6, or 7

2.2.4 CR1-0 Configuration Register (CFG)

Read/Write Power-on Default: 0x00000078

[31:22]	21	[20:17]	16	15	14	13	12	11	10	9	8	7	[6:4]	3	2	1	0
SBZ	nHE	SBZ	STMWA	L4	RAO	V	I	Z	0	R	S	B	SBO	W	C	A	M

This register contains the configuration bits of the FA626TE. Each field is defined below.

Bits[31:22]: Should be zero

Bit 21: Disable hit-under-miss. Default is set to 0.

0 = Enabled

1 = Disabled

Bits[20:17]: Should be zero

Bit 16: STM Write Allocation enable bit

In the write back mode, when STMWA = 1'b1 and RAO = 1'b0, “STM miss” will enable the STM write allocation; if STMWA = 1'b0 and RAO = 1'b0, “STM miss” will not enable the STM write allocation. Programmers must enable the ALO bit of AUX before programming this bit. For FA6265TE5EE0001HC0HA, this bit is not used and should be zero; this bit always performs the write allocation in the write back mode. For FA626TE55EE0001HD0AG, this bit is always set to high and performs the write allocation in the write back mode.

Bit 15: Determine that the T bit is set when the load instruction changes PC.

0 = Load PC to change T bit

1 = Load PC not to change T bit (ARM V4) behavior

Bit 14: Read Allocation Only enable bit

In the write back mode, when RAO = 1'b1, All "STR/STM miss" will not issue the write allocation; when RAO = 1'b0, "STR miss" will issue the write allocation and "STM miss" will depend on the STMWA bit. Programmers must enable the ALO bit of AUX before programming this bit. For FA626TE55EE0001HC0HA, this bit is not used and should be zero, and it always performs the write allocation in the write back mode; For FA626TE55EE0001HD0AG, this bit is always set to low and performs the write allocation in the write back mode.

Bit 13: Vector-based address location. Default is set to '0'.

0 = Vector-based address is set to 0x00000000

1 = Vector-based address is set to 0xFFFF0000

Bit 12: ICache enable bit. Default is set to '0'.

0 = ICache is disabled.

1 = ICache is enabled.

Bit 11: Branch Prediction enabled. Default is cleared to '0'.

0 = Branch prediction is disabled.

1 = Branch prediction is enabled; valid only when MMU is enabled.

Bit 10: Should be zero.

Bit 9: ROM protection bit.

The effect of this bit is described in Section 3.2 "Access Permission Check."

Bit 8: System protection bit.

The effect of this bit is described in Section 3.2.

Bit 7: Big-endian enabled. Default is set to '0'.

0 = Little-endian

1 = Big-endian

Bits[6:4]: Should be one.

Bit 3: Write buffer enabled. Default is set to '1'.

0 = Write buffer is disabled.

1 = Write buffer is enabled; valid only when MMU is enabled.

To enable the write buffer, programmers must set both the write buffer enable and the MMU enable bits. Because MMU is off on reset, the write buffer is disabled by default. When the MMU is turned on, the write buffer will be automatically enabled because the default value of the write buffer enable bit is '1'. Programmers can still disable the write buffer after the MMU is on.

To ensure that the previous data are written out to the external device before the write buffer is disabled, the drain write buffer operation must be performed before the write buffer is disabled. Please refer to the description of the CR7 register for more information about draining the write buffer.

Bit 2: DCache enabled. Default is set to 0.

0 = DCache is disabled.

1 = DCache is enabled; valid only when MMU is enabled.

Bit 1: Alignment check enabled. Default is set to 0.

0 = Alignment check is disabled.

1 = Alignment check is enabled.

Bit 0: Memory Management Unit (MMU) enabled. Default is set to 0.

Special care must be taken when the MMU is turned on or off. Please refer to Section 3.5 for details.

0 = MMU is disabled.

1 = MMU is enabled.

Programmers can access this configuration register through the MRC or MCR instruction. The following instruction is an example of how to access this register:

MCR/MRC p15, 0, Rd, c1, c0, 0 ; Write/Read CFG

2.2.5 CR1-1 Configuration Register (CFG) Auxiliary Control Register (AUX)

Read/Write Power-on default: 0x7

[31:4]	3	2	1	0
Reserved	ALO	SB	DB	RS

The Auxiliary Control Register controls the enable and disable of the branch prediction functions. Each field of this register is defined below:

Bits[31:4]: Reserved

Bit 3: Allocation enable bit. When ALO is enabled, programmers can change the RAO and STMWA bit of CFG. For FA626TE55EE0001HC0HA, this bit is reserved; For FA626TE55EE0001HD0AG, this bit is always low.

Bit 2: Static branch (SB) prediction enable. Default is set to 0.

0 = Disabled

1 = Enabled

Bit 1: Dynamic branch (DB) prediction enabled. Default is set to 0.

0 = Disabled

1 = Enabled

Bit 0: Return Stack (RS) enabled. Default is set to 0.

0 = Disabled

1 = Enabled

Programmers can access this auxiliary control register through the MRC or MCR instruction. The following instruction is an example of how to access this register:

MRC/MRC p15 ,0, Rd, c1, c0, 1; Read/Write the auxiliary control register

2.2.6 CR2 Translation Table Base Register (TTB)

Read/Write Power-on default: 0x0

[31:14]	[13:0]
Translation table base	SBZ

This register defines the base address of the currently active Level-1 page table.

Programmers can access this TTB register through the MRC or MCR instruction. The following instruction is an example to access this register:

MCR/MRC p15, 0, Rd, c2, c0, 0; Write/Read the TTB register

2.2.7 CR3 Domain Access Control Register (DAC)

Read/Write Power-on default: 0x0

[31:30]	[29:28]	[27:26]	[25:24]	[23:22]	[21:20]	[19:18]	[17:16]	[15:14]	[13:12]	[11:10]
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5

[9:8]	[7:6]	[5:4]	[3:2]	[1:0]
D4	D3	D2	D1	D0

This DAC register contains sixteen 2-bit fields; each field defines the access permission for each of the sixteen domains (D15 ~ D0) as shown in Table 2-2 below.

Table 2-2. Domain Access Control Value

Value	Access Type	Description
0b00	No access	Any access generates a domain fault.
0b01	Client	Accesses are checked against the access permission bits in the section or page descriptor.
0b10	Reserved	Has the same effect as No Access.
0b11	Manager	Accesses are not checked against the access permission bits in the section or page descriptor. There, a permission fault cannot be generated.

Programmers can access this DAC register through the MRC or MCR instruction. The following instruction is an example to access this register:

MCR/MRC p15, 0, Rd, c3, c0, 0; Write/Read the DAC register

2.2.8 CR5 Fault Status Register (FSR)

Read/Write Power-on default: 0x0

[31:8]	[7:4]	[3:0]
SBZ	Domain	Status

The Fault Status Register (FSR) stores the cause of the last occurring data or pre-fetched abort and its domain number.

Bits[31:8]: Reserved

Bits[7:4]: Domain

Specify which of the 16 domains D[15:0] has been accessed when a fault occurs.

Bits[3:0]: Status

Indicate the type of access being attempted. The encoding of these bits is shown in Table 2-3.

Table 2-3. Fault Priority and FSR Encoding

Priority	Source		FS[3:0]	Domain[3:0]	FAR	Note
Highest	Alignment	-	0b0001	Invalid	Valid	DFSR only
Second to highest	External abort on translation	Section	0b1100	Invalid	Valid	-
		Page	0b1110	Valid	Valid	
Third to highest	Translation	Section	0b0101	Invalid	Valid	-
		Page	0b0111	Valid	Valid	
Forth to highest	Domain	Section	0b1001	Valid	Valid	-
		Page	0b1011	Valid	Valid	
Fifth to highest	Permission	Section	0b1101	Valid	Valid	-
		Page	0b1111	Valid	Valid	
Next to lowest	Lock abort This data abort occurs on an MMU lock operation	-	0b0110	Invalid	Valid	DFSR only

Priority	Source		FS[3:0]	Domain[3:0]	FAR	Note
Lowest	External abort	Section	0b1000	Valid	Valid	-
		Page	0b1010	Valid	Valid	

Programmers can use the following instructions to access the data and pre-fetch FSR:

MRC p15, 0, Rd, c5, c0, 0	; Read data FSR
MCR p15, 0, Rd, c5, c0, 0	; Write data FSR
MRC p15, 0, Rd, c5, c0, 1	; Read pre-fetch FSR
MCR p15, 0, Rd, c5, c0, 1	; Write pre-fetch FSR

Notes:

1. The MMU lock operation does not check the alignment, domain, and permission faults.
2. The invalidate ICache/DCache entry by using the MVA instruction or the pre-fetched ICache entry instruction will signal the data abort when a translation error or an external abort on the translation occurs. These cause the updates of DFSR and DFAR.

2.2.9 CR6 Fault Address Register (FAR)

Read/Write Power-on default: 0x0

[31:0]
Fault address

This register holds the Modified Virtual Address (MVA) of the last occurring data fault. The FAR is only updated for data faults. Programmers can access FAR through the MRC or MCR instruction. The following instruction is an example of how to access this register:

MCR/MRC p15*, 0, Rd, c6, c0, 0* ; Write/Read Data Abort FAR register

2.2.10 CR7 Cache Operation Register (Write only)

The Cache Operation Register defines the cache management operations. Writing this register invokes corresponding instructions as defined in Table 2-4. Reading this register can generate unpredictable results.

Table 2-4. CP15 Cache Control Instructions

Operation	Instruction Format	Rd
Instruction cache operation		
Invalidate ICache All	MCR p15, 0, Rd, C7, C5, 0	Should be zero
Invalidate ICache Entry (Using MVA)	MCR p15, 0, Rd, C7, C5, 1	MVA
Invalidate ICache Entry (Using Index)	MCR p15, 0, Rd, C7, C5, 2	Set/Index
Pre-fetch ICache Entry	MCR p15, 0, Rd, C7, C13, 1	MVA
Prefetch Flush	MCR p15, 0, Rd, C7, C5, 4	Should be zero
Data cache operation		
Invalidate DCache All	MCR p15, 0, Rd, C7, C6, 0	Should be zero
Invalidate DCache Entry (Using MVA)	MCR p15, 0, Rd, C7, C6, 1	MVA
Invalidate DCache Entry (Using Index)	MCR p15, 0, Rd, C7, C6, 2	Set/Index
Clean DCache All	MCR p15, 0, Rd, C7, C10, 0	Should be zero
Clean DCache Entry (Using MVA)	MCR p15, 0, Rd, C7, C10, 1	MVA
Clean DCache Entry (Using Index)	MCR p15, 0, Rd, C7, C10, 2	Set/Index
Clean and Invalidate DCache All	MCR p15, 0, Rd, C7, C14, 0	Should be zero
Clean and Invalidate DCache Entry (Using MVA)	MCR p15, 0, Rd, C7, C14, 1	MVA
Clean and Invalidate DCache Entry (Using Index)	MCR p15, 0, Rd, C7, C14, 2	MVA
Invalidate ICache and DCache All	MCR p15, 0, Rd, C7, C7, 0	Should be zero
Write buffer operation		
SYNC (Drain write buffer)	MCR p15, 0, Rd, C7, C10, 4	Should be zero
Power-down operation		
Wait for Interrupt (Idle mode)	MCR p15, 0, Rd, C7, C0, 4	Should be zero
Branch Target Buffer operation		
Invalidate BTB All	MCR p15, 0, Rd, C7, C5, 6	Should be zero
Instruction scratchpad RAM operation		
Invalidate IScratchpad All	MCR p15, 0, Rd, C7, C5, 5	Should be zero

The format of Modified Virtual Address (MVA) is shown below. Bits[31:5] are the MVA of the cache line that users want to invalidate or clean.

Table 2-5. **Modified Virtual Address Format**

[31:5]	[4:0]
Modified virtual address	Ignored

The format of Way/Index is shown below. Bits[31:30] are the Way of the cache line that users want to invalidate or clean. Bits[12:5] are the Index in which users want to invalidate or clean the cache line.

Table 2-6. **Way/Index Format**

[31:30]	[30:13]	[12:5]	[4:0]
Way	SBZ	Index	SBZ

Each operation is explained below:

Invalidate DCache / ICache Entry

The Invalidated DCache/ICache entry operation invalidates the cache line (Clear the valid bit) specified in Rd regardless of the line is dirty or not. The content of the line will not be written back to the memory even if it is dirty.

Clean DCache Entry

The Clean DCache entry operation will write back the content of the cache line specified in Rd to the memory if the line is dirty. The line remains in cache with the dirty bit cleared.

Clean and Invalidate DCache Entry

If a cache line is dirty, the clean and invalidate DCache entry operation will write back the cache line content specified in Rd to the memory and will clear the dirty and valid bits. If the cache line is not dirty, this operation will clear the valid bit directly.

SYNC

This operation enforces the order of the memory accesses. Once it is executed, the on-going data fetch for the previous miss (This happens because the hit-under-miss cache accessing FA626TE) has to be completed and the write buffer has to be drained before executing the following instructions.

Prefetch Flush

This operation enforces an instruction memory barrier in the program sequence. If PA is different from VA after turning on MMU, this operation can be used right after the instruction that turns on MMU to ensure that the subsequent instructions will be fetched with translation.

Wait for Interrupt

This operation forces FA626TE to enter a low-power mode. In this mode, the processor is halted in an idle state until an IRQ, FIQ, or external ICE event happens.

Invalidate BTB All

This instruction invalidates all the entries in Branch Target Buffer (BTB).

Invalidate IScratchpad All

This instruction clears the valid bit of instruction scratchpad RAM so that it can be automatically filled again after re-defining the region or size of the instruction scratchpad.

Compatibility Issues

Some efforts on porting the cache management instructions are needed for software that is developed for a different ARM core variant.

2.2.11 CR8 TLB Operation Register (Write only)

The TLB operation register defines the TLB management operations. Writing this register will invoke the corresponding operations as defined in the following table. Reading this register will cause unpredictable results. UTLB refers to the unified TLB in the FA626TE design. The invalidate ITLB and DTLB operations are also provided as equivalent UTLB operations. Executing either operation invalidates the unified TLB.

Table 2-7. TLB Operations

Operation	Instruction Format	Rd
Invalidate UTLB All	MCR p15, 0, Rd, C8, C7, 0	Should be zero
Invalidate UTLB Entry	MCR p15, 0, Rd, C8, C7, 1	MVA
Invalidate ITLB All	MCR p15, 0, Rd, C8, C5, 0	Should be zero
Invalidate ITLB Entry	MCR p15, 0, Rd, C8, C5, 1	MVA
Invalidate DTLB All	MCR p15, 0, Rd, C8, C6, 0	Should be zero
Invalidate DTLB Entry	MCR p15, 0, Rd, C8, C6, 1	MVA

The locked entries are protected from “Invalidate UTLB All.” These entries can only be cleared by “Invalidate UTLB Entry”. Please note that executing “Translate and Lock UTLB Entry” when MMU is not enabled has no effect on the FA626TE cores. Therefore, this operation can be ignored.

2.2.12 CR9_0 Data/Instruction Cache Lockdown Register (DCL/ICL)

Read/Write Power-on default: 0x0

31	[30:0]
Lock	Reserved

This register defines the data/instruction cache lockdown operation. Writing bit 31 of CR9 will invoke the cache lockdown operation. Please refer to Chapter 4 for more information on the cache lockdown operation.

Bit 31: Lock bit. Default is set to ‘0’.

0: No effect

1: Invoke cache lockdown operation

Bits[30:0]: Reserved

Programmers can access the cache lockdown register through the MRC or MCR instruction. The following instructions are examples to access these registers. Programmers should note that the opcode_2 field of these instructions must be ‘0’ to access the DCL registers and must be ‘1’ to access the ICL registers.

```
MCR/MRC p15, 0, Rd, c9, c0, 0        ; Write/Read DCL
MCR/MRC p15, 0, Rd, c9, c0, 1        ; Write/Read ICL
```

Notes:

1. For DCache lockdown, programmers must guarantee that the fetched cache line is cache missed. Otherwise, the lockdown action will be ignored.
2. For ICache pre-fetch and lockdown, programmers must guarantee that the line address being locked down does not overlap with the scratchpad.

2.2.13 CR9_1 Data/Instruction Scratchpad Configuration Register (DSC/ISC)

Read/Write Power-on default: {32'h00000000}

[31:10]	[9:6]	[5:2]	1	0
DSB/ISB	SBZ	DSS/ISS	0/1	DSE/ISE

This register defines the instruction scratchpad region and data scratchpad region in the memory. On reset, the scratchpad enable bit (ISE/DSE) is cleared to '0', and all other bits are undefined. It is necessary to initialize bits[31:1] before or when the scratchpad is enabled (ISE/DSE = '1'). The setting of the scratchpad is valid only when MMU is on. Each field is described below:

Bits[31:10]: Scratchpad Base (DSB/ISB)

DSB and ISB define the base address (MSBs) of the instruction scratchpad and the data scratchpad, respectively. Hardware will automatically align the base address to the size boundary defined by DSS (For data) or ISS (For instruction). The value set to DSB or ISB is in bits[31:10] of a physical address.

Bits[5:2]: Scratchpad Size (DSS/ISS)

DSS and ISS define the size of the instruction scratchpad and the data scratchpad, respectively. Table 2-8 shows the encoding.

Table 2-8. Definitions of Scratchpad Sizes

DSS/ISS	Region-x Size
0 0 0 0	Reserved
0 0 0 1	1 kB
0 0 1 0	2 kB
0 0 1 1	4 kB
0 1 0 0	8 kB
0 1 0 1	16 kB
0 1 1 0	32 kB
0 1 1 1	64 kB
1 0 0 0	128 kB
1 0 0 1 ~ 1 1 1 1	Reserved

Bit 1: Self-loading capability

After reset, this bit is set to '1' for the instruction scratchpad and is set to '0' for the data scratchpad.

Bit 0: Enable data/instruction scratchpad RAM

Setting this bit enables the DScratchpad/IScratchpad region.

Programmers can access this data/instruction scratchpad configuration register through the MRC or MCR instruction. The following instructions are examples to access these registers. The opcode_2 field of these instructions is set to '0' for accessing the data scratchpad configuration register and is set to '1' for accessing the instruction scratchpad configuration register.

MCR/MRC p15, 0, Rd, c9, c1, 0	Write/Read the data scratchpad configuration register
MCR/MRC p15, 0, Rd, c9, c1, 1	Write/Read the instruction scratchpad configuration register

2.2.14 CR10 TLB Lockdown Register (Write only)

The TLB lockdown register defines the TLB lockdown operations. Writing this register will invoke the TLB lockdown operations as defined in the following table. Reading this register will cause an unpredictable result. UTLB refers to the unified TLB in the FA626TE design.

Table 2-9. TLB Lockdown Operations

Operation	Instruction Format	Rd
Translate and Lock UTLB Entry	MCR p15, 0, Rd, C10, C8, 0	MVA
Unlock UTLB All	MCR p15, 0, Rd, C10, C8, 1	Should be zero

The operation, "Translate and Lock UTLB Entry", does not imply checking the domain and AP permission. If a translation abort is generated during this operation, the data fault status register will be updated to indicate a "Lock Abort". The exception is reported as a data abort no matter it is induced by an instruction or data fetch. Please note that executing "Translate and Lock UTLB Entry" when MMU is not enabled has no effect on the FA626TE core. Therefore, this operation can be ignored.

2.2.15 CR13 Process ID (PID) Register

Read/Write Power-on default: 0x0

[31:25]	[24:0]
PID	UNP/SBZP

To avoid the address aliasing problem and achieve fast context switch, the virtual address in FA626TE cores is translated to a Modified Virtual Address (MVA) according to the PID value. The translation is shown below:

```
If (VA[31:25] = 0b0000000) , then
MVA = VA | (PID << 25)
else
MVA = VA
```

Programmers can access the PID register through the MRC or MCR instruction. The following instruction is an example of how to access this register:

```
MCR/MRC p15, 0, Rd, c13, c0, 0 ; Write/Read the process ID register
```

2.2.16 CR15_2 Peripheral Port Region Definition Register

Read/Write Power-on default: 0x0

[31:12]	[11:5]	[4:0]
Base address	UNP/SBZ	Size

The peripheral port region definition register determines whether a non-cacheable data access can go through the peripheral port. When the physical address is located in the peripheral port region, data access will go through the peripheral port. Otherwise, the data access will go through the Data Read (DR) or Data Write (DW) port. Please note that only the non-cacheable access can go through the peripheral port; the cacheable access will still be carried out through the DR or DW port.

Bits[4:0]: Size

Define the size of the peripheral port region. Table 2-10 shows the encoding.

Table 2-10. Definitions of Region Sizes

Size Field	Region Size
0 0 0 0 0	0 kB
0 0 0 1 1	4 kB
0 0 1 0 0	8 kB
0 0 1 0 1	16 kB
0 0 1 1 0	32 kB
0 0 1 1 1	64 kB
0 1 0 0 0	128 kB
0 1 0 0 1	256 kB
0 1 0 1 0	512 kB
0 1 0 1 1	1 MB
0 1 1 0 0	2 MB
0 1 1 0 1	4 MB
0 1 1 1 0	8 MB
0 1 1 1 1	16 MB
1 0 0 0 0	32 MB
1 0 0 0 1	64 MB
1 0 0 1 0	128 MB
1 0 0 1 1	256 MB
1 0 1 0 0	512 MB
1 0 1 0 1	1 GB
1 0 1 1 0	2 GB

Bits[31:12]: Base Address

Define the physical base address of the peripheral port region

Programmers can access the peripheral port definition register through the MRC or MCR instruction. The following example shows how to access this register:

MCR/MRC p15, 0, Rd, c15, c2, 4; Write/Read the peripheral port region definition register

2.2.17 CR15 Test Register (Write only)

This register is used to access the TLB content for debugging. Please refer to Appendix D for details.

2.3 Software Breakpoint (BKPT)

The Breakpoint (BKPT) instruction causes a software breakpoint. In FA626TE, the debug hardware is included and will always override the normal behavior. The normal behavior will enter the pre-fetch abort and handle the breakpoint itself. When the BKPT instruction is executed, the hardware will enter the ICE mode at the BKPT instruction.

Chapter 3

Memory Management Unit (MMU)

This chapter contains the following sections:

3.1 General Description

FA626TE implements an enhanced ARM architecture, V5-compatible MMU, to provide the address translation and access the permission check for instruction fetch and data access. The FA626TE Memory Management Unit (MMU) has two levels of TLB structure. The level-1 TLB structure includes an ITLB for the instruction access and a DTLB for the data access. Both accesses are 8-entry fully-associative. The level-2 TLB (UTLB) structure has a unified 64-entry 2-way set-associative structure. The FA626TE MMU searches the level-1 TLB first. If the page table information is not found, the FA626TE MMU will then search for UTLB. The external memory access will be issued to read the page table information in the main memory if the level-1 TLB and level-2 TLB structures are not found. All the search processes are completed by using the page-walking hardware.

FA626TE MMU has the following features:

- ARM architecture V5 MMU address translation and access permission check scheme
- Supports invalidation of all TLB entries with a single instruction
- Supports invalidation of a single TLB entry
- Supports TLB translation and lockdown function

3.2 MMU Program Accessible Registers

Table 3-1 lists the CP15 registers related to the MMU function. These registers cooperate with the page descriptors stored in the page translation table to determine the operation of the MMU. Please refer to Section 2.2 for detailed register definitions.

Table 3-1. MMU Program Accessible Registers

Register	Corresponding Register	Bit Location	Description
Control register	CR1	M, A, S, R	M: MMU enable bit A: Alignment fault check enable bit S: System protection bit R: ROM protection bit
Translation table base register	CR2	[31:14]	Contain the base address of the active Level-1 page table
Domain access control register	CR3	[31:0]	Contain 16 2-bit fields, each field defines the access control attribute for one of the 16 domains (D15 ~ D0).
Fault status register	CR5 (I and D)	[7:0]	Hold the status of the last occurring data or pre-fetched abort and its domain number

Register	Corresponding Register	Bit Location	Descriprtion
Fault address register	CR6 (D only)	[31:0]	Hold the Modified Virtual Address (MVA) which causes the last occurring data abort Users can access register CP14 for the fault address of a pre-fetch abort.
TLB operation register	CR8 (Unified)	[31:0]	Writing to this register induces TLB management operations.

3.3 Address Translation

The FA626TE MMU supports the standard ARM V5 MMU address translation. The physical addresses are translated automatically by hardware from Modified Virtual Addresses (MVA), issued from the FA626TE core after the PID translation. The FA626TE MMU contains the Translation Lookaside Buffer (TLB) to keep the most recently used translation informaton, including the physical address and access permission. The page table walk process can be divided into two stages depending on whether the access is section-mapped or page-mapped. The first stage fetches a level-one descriptor and the second stage fetches a Level-2 descriptor. The section-mapped access needs only a level-one translation and the page-mapped access needs a Level-1 translation followed by a Level-2 descriptor fetch.

3.3.1 Translation Page Table

The hardware table walk is issued when the TLB does not contain information for the currently requested MVA. The TTB (Please refer to the translation table base register defined in Section 2.2.) points to the base address of the translation page table, including both the Level-1 and Level-2 descriptors, in the physical memory. The section-mapped access only comprises the 1-M-size section type, whereas the page-mapped access comprises three types: 64K-size large page, 4K-size small page, and 1K-size tiny page. The smallest possible table for the full 4-G address memory space is constructed of 4096 entries (4K-word memory space) 1-M-size section descriptors for the translation page table. Figure 3-1 shows the page table walk process.

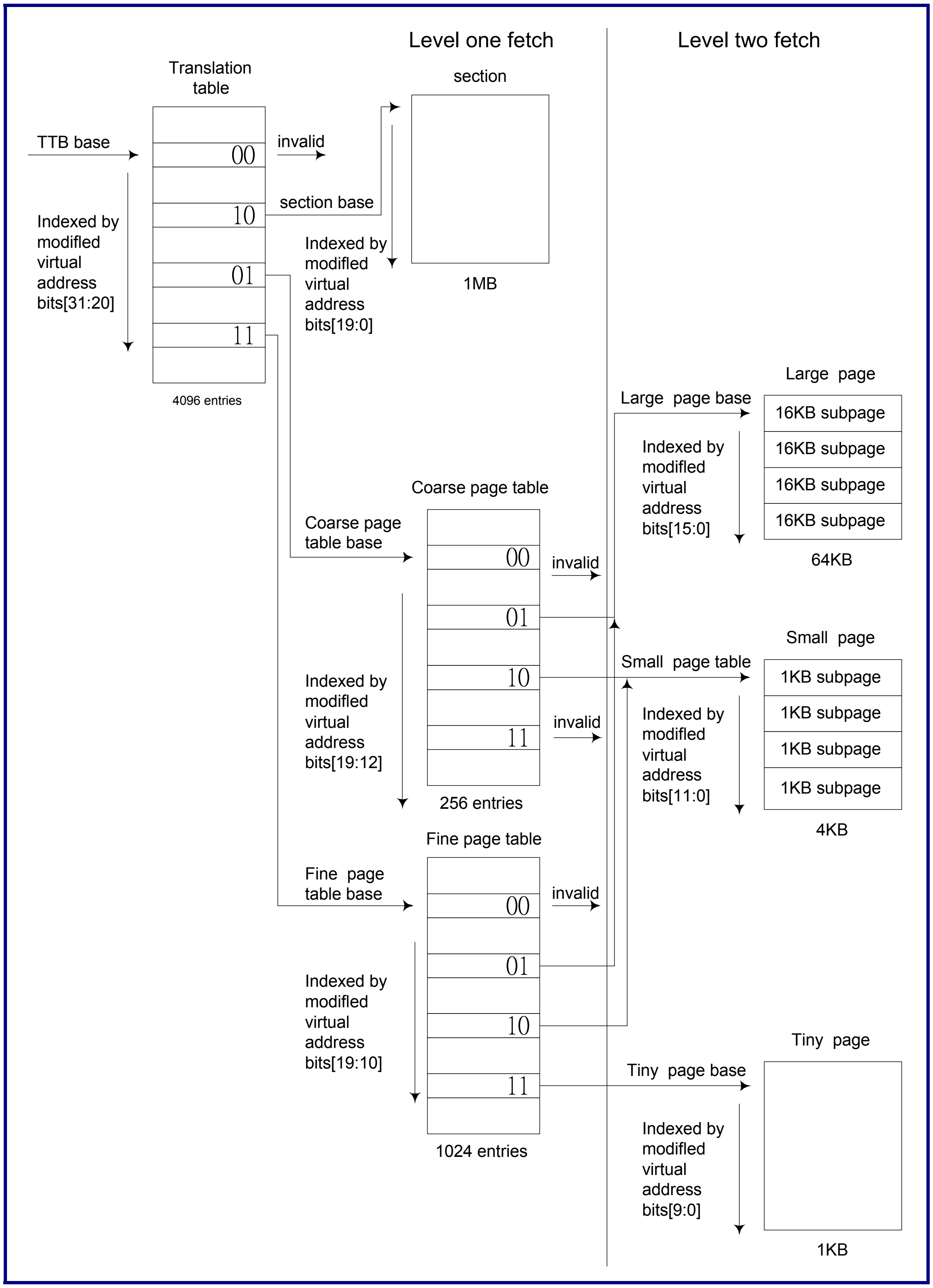

Figure 3-1. Table Walking Process

3.3.2 Level-1 Translation

The harware translation process always starts out by the Level-1 descriptor fetch. Figure 3-2 shows the Level-1 fetch process flow.

3.3.2.1 Level-1 Fetch

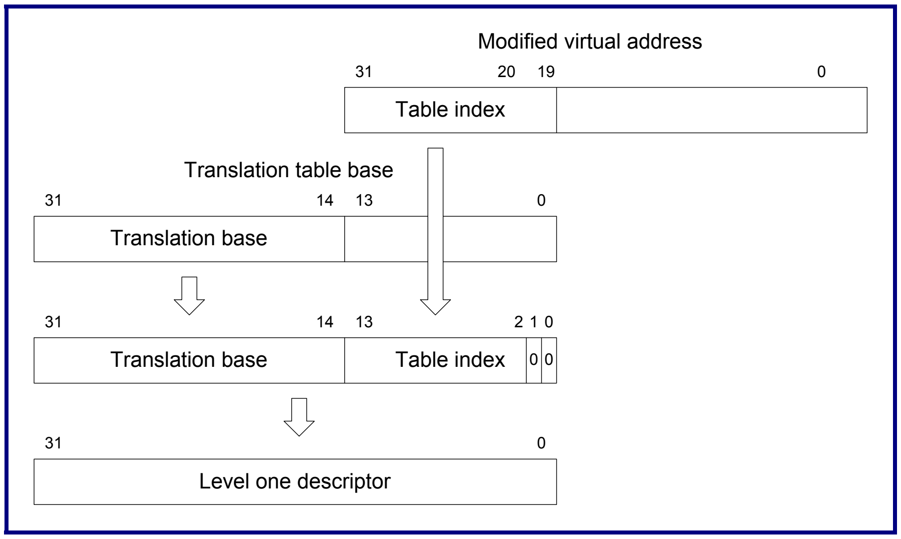

Figure 3-2. Level-1 Descriptor Fetch

3.3.2.2 Level-1 Descriptor

Table 3-2. Level-1 Descriptor

Type/Bit	[31:20]	[19:12]	[11:10]	9	[8:5]	4	3	2	1	0
Fault									0	0
Coarse page table	Coarse page base address			0	Domain	1	0	0	0	1
Section	Section base address	SBZ	AP	0	Domain	1	C	B	1	0
Fine page table	Fine page base address		SBZ'		Domain	1	0	0	1	1

Coarse page table

Bits[31:10]: These bits come form the base address of the coarse page table for Level-2 translation.

Bit 9: Should be zero

Bits[8:5]: Domain control bits

Bit 4: Should be one

Bits[3:2]: Should be zero

Bits[1:0]: Level-1 page type

‘01’ =I Indicates the coarse-type level-1 page and must issue another level-2 translation

Section

Bits[31:20]: These bits come form the physical address for a section.

Bits[19:12]: Should be zero

Bits[11:10]: Access permission for permission check. Please refer to Section 3.4.4 for detailed explanation.

Bit 9: Should be zero

Bits[8:5]: Domain control bits

Bit 4: Should be one

Bit 3: Defines this page as cacheable

Bit 2: Defines this page as bufferable

Bits[1:0]: Level-one page type

‘10’ =Indicates the section-type page

Fine page table

Bits[31:12]: These bits come form the base address of the fine page table for level-2 translation.

Bits[11:9]: Should be zero

Bits[8:5]: Domain control bits

Bit 4: Should be one

Bits[1:0]: Level-1 page type

‘11’ = Indicates the fine-type Level-1 page and must issue another Level-2 translation

3.3.3 Level-2 Translation

The page-mapped access will continue with Level-2 descriptor fetch if the Level-1 tranalation returns coarse or fine page table. The coarse or fine page table provides the base address of Level-2 desciptor in physical memory, because the L2 table index overlaps with the page table index under some translation conditions. Table 3-3 depicts these conditions and special concerns about the Level-2 descriptor.

Table 3-3. Special Concerns of Level-2 Descriptor

Translation Direction	Special Concern
Large page translated from coarse page	Level-2 descriptor repeated in 16 consecutive entries
Small page translated from coarse page	Level-2 descriptor repeated in each entry
Tiny page translated from coarse page	Invalid
Large page translated from fine page	Level-2 descriptor repeated in 64 consecutive entries
Small page translated from fine page	Level-2 descriptor repeated in 4 consecutive entries
Tiny page translated from fine page	Level-2 descriptor repeated in each entry

3.3.3.1 Level-2 Descriptor

Type/Bit	[31:16]	[15:12]	[11:10]	[9:8]	[7:6]	[5:4]	3	2	1	0
Fault									0	0
Large page	Large page base address	SBZ	AP3	AP2	AP1	AP0	C	B	0	1
Small page	Small page base address		AP3	AP2	AP1	AP0	C	B	1	0
Tiny page	Tiny page base address			SBZ		AP	C	B	1	1

Large page

Bits[31:16]: These bits come form the physical address for a large page.

Bits[15:12]: Should be zero

Bits[11:4]: Access permission for the corresponding sub-page

Bit 3: Defines this page as cacheable

Bit 2: Defines this page as bufferable

Bits[1:0]: Level-one page type

‘01’ = Indicates the large page

Small page table

Bits[31:12]: These bits come form the physical address for a small page.

Bits[11:4]: Access permission for the corresponding subpage

Bit 3: Defines this page as cacheable

Bit 2: Defines this page as bufferable

Bits[1:0]: Level-1 page type

‘10’ = Indicates the small page

Tiny page table

Bits[31:10]: These bits come form the physical address for a tiny page.

Bits[9:6]: Should be zero

Bits[5:4]: Access permission

The tiny page does not support sub-pages.

Bit 3: Defines this page as cacheable

Bit 2: Defines this page as bufferable

Bits[1:0]: Level-1 page type

‘11’ = Indicates the tiny page

3.3.4 Translation Flow References

In the following sub-sections, Some example cases of the hardware translation sequence are addressed.

3.3.4.1 Section Translation Flow

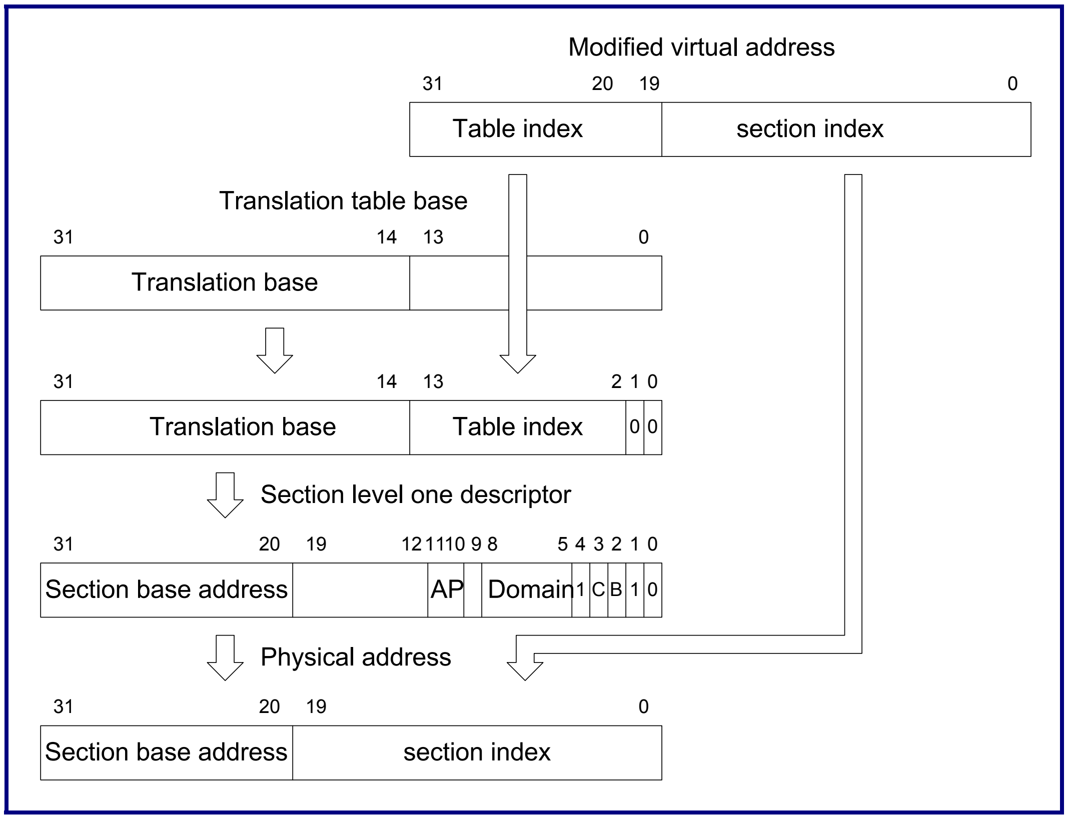

Figure 3-3. **Section Translation Flow**

3.3.4.2 Translating Large-page from Coarse Page Table

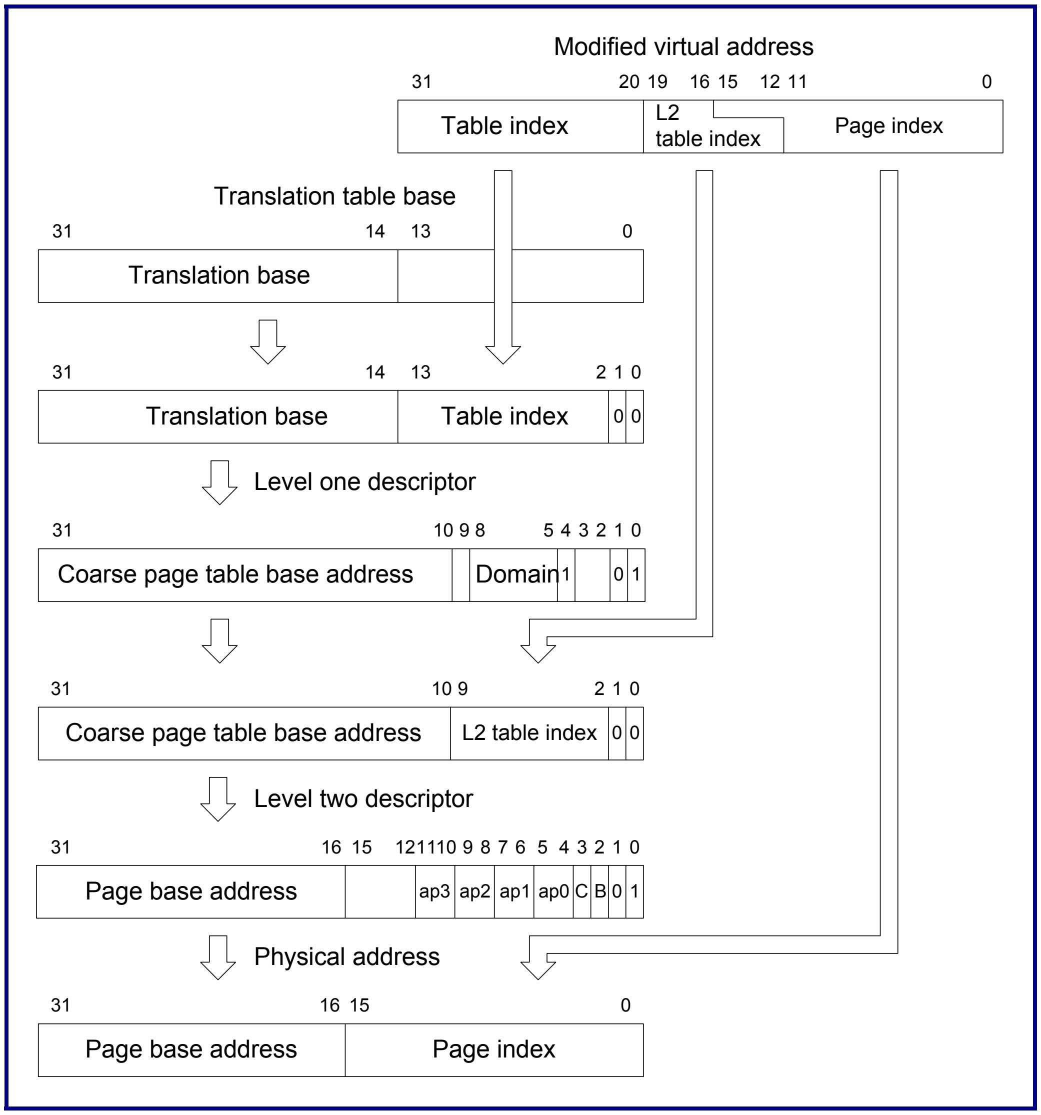

Figure 3-4. Translating Large-page from Coarse Page Table

3.3.4.3 Translating Small-page from Coarse Page Table

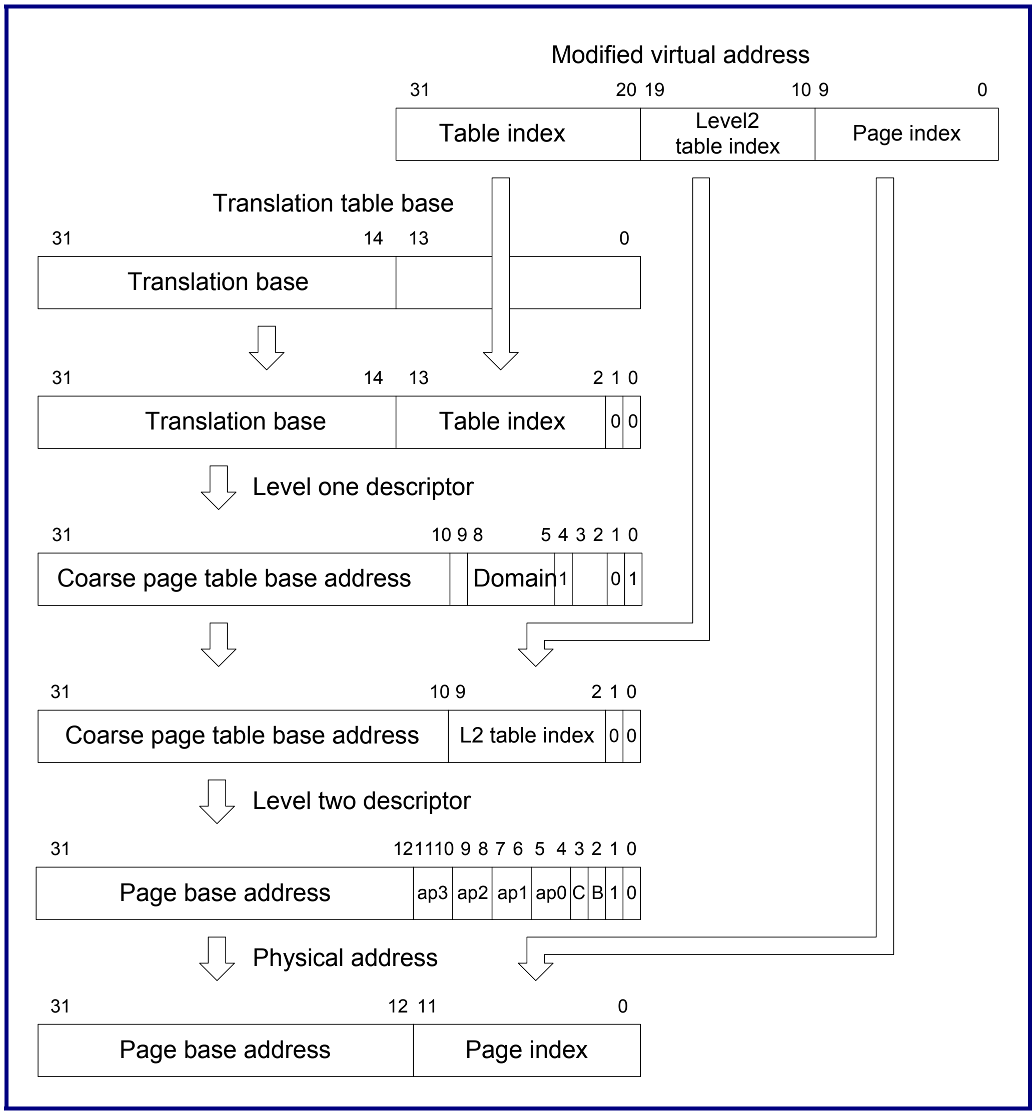

Figure 3-5. Translating Small-page from Coarse Page Table

3.3.4.4 Translating Tiny-page from Fine Page Table

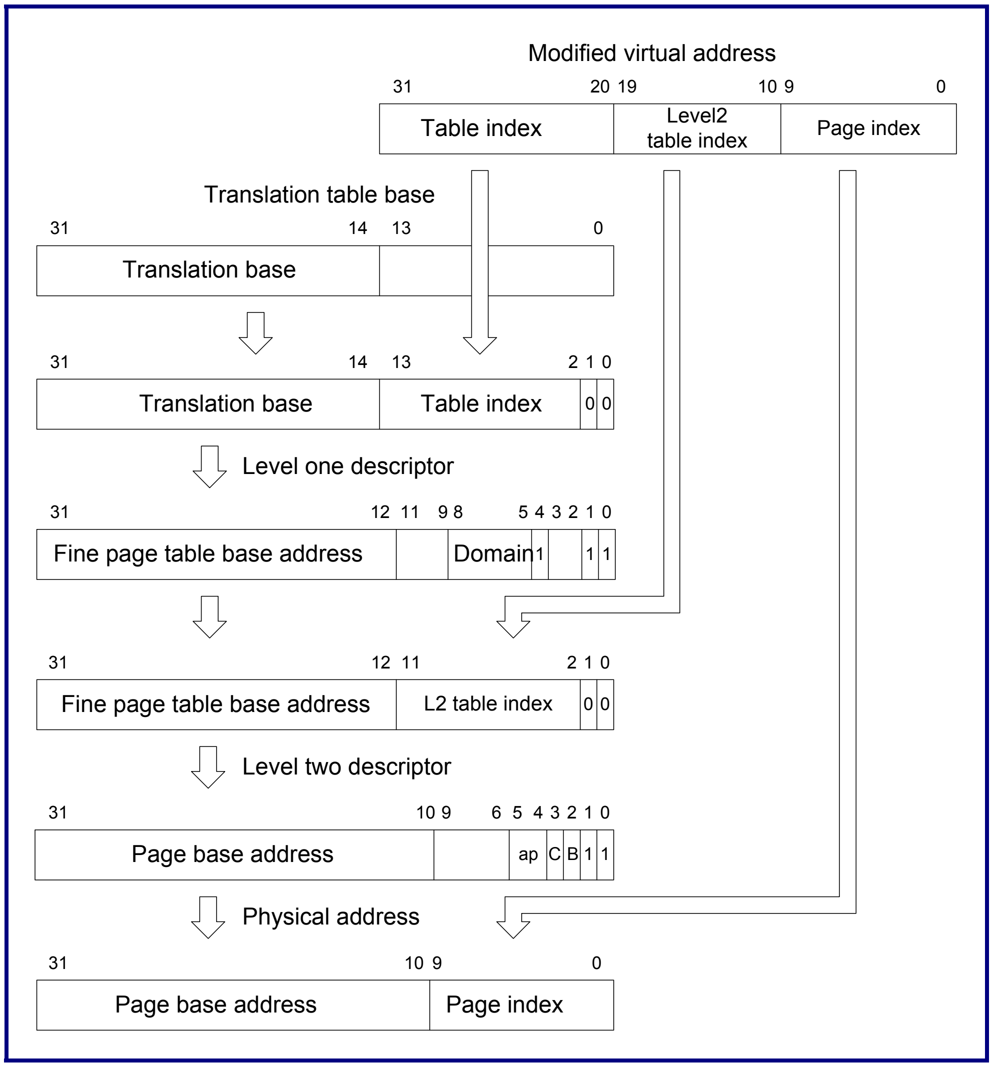

Figure 3-6. Translating Tiny-page from Fine Page Table

3.3.5 Sub-pages

The FA626TE MMU supports sub-page for both large and small pages. It defines four sub-pages that own individual applications. The applications of the sub-pages play the same role as a whole AP. FA626TE TLB will keep four individual APs in a single entry even when sub-pages exist. This means that users cannot invalidate only one sub-page by using the invalidate UTLB entry operation.

3.3.6 Pre-fetch Effect

FA626TE incorporates a Branch Target Buffer (BTB). Because BTB stores the PC and target addresses of the branch instructions, it is not necessary to flush BTB on a context switch. However, programmers can choose to flush BTB depending on the performance consideration.

3.4 Fault Check Sequence

The fault check sequence used in the FA626TE MMU is shown in Figure 3-7. In the following section, the sequence will be discussed in details.

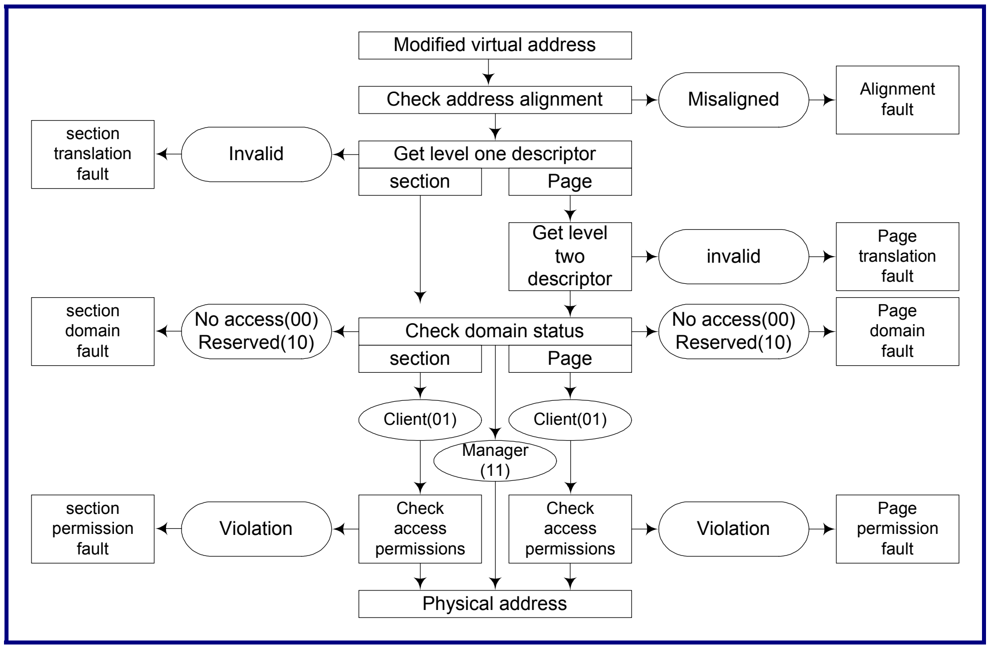

Figure 3-7. Fault Check Sequence

3.4.1 Alignment Fault

The first step of the fault check aligns with the fault check when the Alignment Check (Bit 1 of the CP15 register 1) is enabled. The alignment fault emerges under the following cases, irrespective of whether the MMU is enabled or not.

- Any data word access performed with the address is not word-aligned.
- Any data half-word access performed with the address is not half-word aligned.

3.4.2 Translation Fault

The FA626TE MMU generates the following two translation faults:

- Section

  It will be generated if the returned level-1 descriptor is invalid when bits[1:0] are ‘0’.

- Page

  It will be generated if the returned Level-2 descriptor is invalid when bits[1:0] are ‘0’, or translated the level-2 tiny page coming from the level-1 coarse page table.

3.4.3 Domain Fault

The FA626TE MMU generates the following two domain faults:

- Section

  The level-1 descriptor includes a specified file and domain. This descriptor holds four bits to select one of the 16 2-bit data streams in the domain access control register. If the 2-bit data stream is set to ‘0’ or ‘10’, a domain fault will be generated.

- Page

  The page domain fault check is the same as the section domain fault check, but FSR will be encoded as 0x1011 to distinguish the domain fault generated at the page-mapped access.

The domain access control is a scheme to control the access right of a group of pages or sections. A maximum of 16 domains, each with an individual domain access control value, can be defined in the CP15 register 3.

3.4.4 Permission Fault

The access permission check can be performed through the Domain Access Control (DAC) register, accessing the permission bits of a page, the S (System mode) and R (ROM protection) bit in the CP15 register 1, and the user/supervisor mode bit. If the access control value of the specified domain is set to Client, its access permission is checked according to Table 3-4.

Table 3-4. Permission Check

AP	S	R	Supervisor Permission	User Permission	Description
00	0	0	No access	No access	Any access generates a permission fault.
00	1	0	Read only	No access	Only Supervisor read is permitted.
00	0	1	Read only	Read only	Any write generates a permission fault.
00	1	1	-	-	Reserved
01	x	x	R/W	No access	Access is allowed only in supervisor mode.
10	x	x	R/W	Read only	Write in User mode causes permission fault.
11	x	x	R/W	Read / Write	All access types are permitted in both modes.

3.5 Enabling MMU

Please follow the sequence below to enable MMU:

1. Prepare the page table in a memory
2. Program the TTB and domain access control registers
3. Enable MMU by setting bit 0 of the control register
4. Branch to MVA or execute the PrefetchFlush instruction

Users should pay more attention on the MMU enabling sequence if the address mapping of instructions is changed after the MMU is turned on. A sample of the MMU enabling sequence is shown below to explain the effect:

```
MRC     p15, 0, R1, c1, c0, 0; Read the control register
ORR     R1, #1
MCR     p15, 0, R1, c1, c0, 0; Enable MMU
(Fetch flat)
(Fetch flat)
(Fetch flat)
(Fetch flat)
(Fetch translated)
```

PrefetchFlush can be used to ensure that all subsequent instructions are fetched with translation.

```
MCR     p15, 0, R1, c1, c0, 0; Enable MMU
```

MCR p15, 0, Rd, c7, c5, 4; PrefetchFlush instruction
(Fetch translated)

Users can simultaneously enable ICache, DCache, and MMU by using one MCR instruction.

3.6 Disabling MMU

Clearing bit 0 in the control register will simultaneously disable MMU and data cache. Similar to the case of turning on MMU, four instructions can be executed after the instruction disables MMU translating with the original mapping.

Chapter 4

Cache and Write Buffer

This chapter contains the following sections:

FA626TE applies a Harvard architecture design with separated instruction and data caches. The instruction cache (ICache) and the data cache (DCache) implements the 4-way set-associate architecture. The cache sizes can be configured to 4 kB, 8 kB, 16 kB, 32 kB, or 64 kB. Each cache line contains eight 32-bit words. The cache organization is virtually-indexed and physically-tagged in FA626TE. The 32-kB cache architecture is shown in Figure 4-1. The FA626TE caches use the LRU algorithm for the cache line replacement.

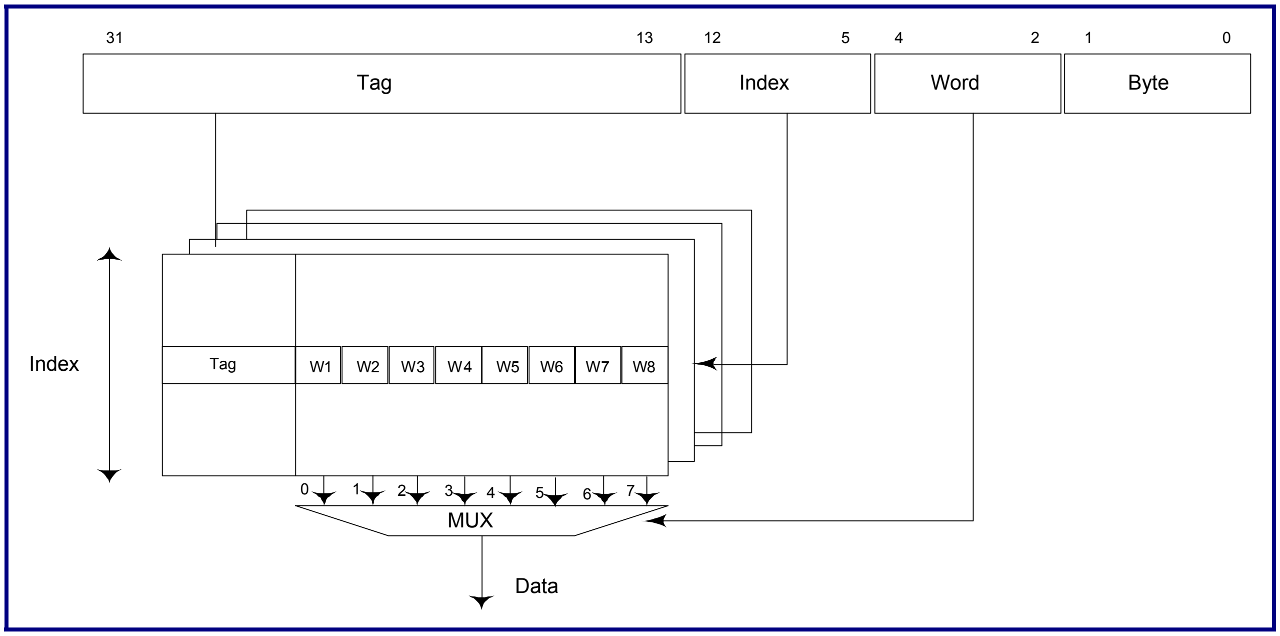

Figure 4-1. Cache Organization

Each cache can be separated into two parts: The tag memory and the data memory. The tag memory stores the tag address of cached data, while the data memory stores the cached data itself. For a given virtual address (Based on the example of 32 kB), bit 12 to bit 5 of the virtual address are referred as (Virtual) “index”, while bit 31 to bit 10 of the physical address are referred as (Physical) “tag”. To access data from cache, the virtual index is used to address an entry in the tag memory. The tag outputs, each of the four ways from the tag memory, are compared with the tag portion of the data physical address. If a match (Called a cache hit) is found, the accessed data from cache is sent to the processor core. If there is no match (Called a cache miss), data need to be fetched from an external memory.

Because the FA626TE cache is virtually-indexed and physically-tagged, when the cache size is greater than the smallest page size, the high order bits of the (Virtual) index will be translated by TLB. As a result, the virtual index may be different from its corresponding physical index. If a system allows more than one virtual address to be mapped to the same physical address, the consistency of such synonyms can be guaranteed by the following software mechanisms:

1. Allow only one of the synonyms to exist from the cache by cleaning and invalidating the current synonym line from the cache before the next synonym is brought in; or
2. Use the memory management routine to make sure that a shared page uses the same high order bits for the virtual addresses. By setting the size of 8 kB and the smallest page size of 1 kB, the high order bits are bit 12 to bit 10.

The FA626TE data cache features the non-blocking (Hit-under-miss) cache access. The non-blocking cache access is implemented to reduce the performance loss resulting from cache miss. When a data access is missed in a cache, the following instructions can still be proceed if these instructions have no data dependence with the missed data and induce no further cache miss. Consequently, the pipeline will not need to be stalled for the long access time required to get the missed data. This significantly improves the processor performance.

FA626TE also features the cache lockdown function. Programmers can lock down the performance-critical code or frequently used data in cache to prevent them from being evicted on the cache line replacement. Locked lines will not be replaced and will remain in the cache until explicitly invalidated through cache management operations. This feature is especially useful in embedded applications. When not all ways of an index are locked, the lock operation finds a way to be replaced (For lock) using the normal replacement algorithm. If all ways of an index are locked, further lockdown to the same index will unlock and replace way 0, and lock with the new line.

ICache and DCache are enabled through programming coprocessor 15 (CP15), the system control processor. For register definition and complete description on CP15, please refer to Chapter 2 and Chapter 3 for more details. Please note that the translation table and related attributes must be programmed before enabling MMU. The cache management operations are also provided through CP15 and are introduced in the following sub-sections.

4.1 Instruction Cache (ICache)

ICache is disabled by default. Programmers need to set bit 12 of the CR1 register in CP15 to enable ICache. The complete ICache contents are invalidated on reset.

FA626TE provides the following management operations for ICache:

1. Invalidate entire ICache
2. Invalidate a single ICache line using MVA
3. Invalidate a single ICache line using set/index
4. Pre-fetch an ICache line
5. Lock down an ICache line

The operations are described below. The invalidation operations take effect even when ICache is turned off. All cache management operations have to be executed in the supervisor mode.

Invalidate entire ICache

Programmers can invalidate the entire ICache through the following system coprocessor instruction.

```
MCR 15, 0, Rd, C7, C5, 0
```

The value in the Rd register should be zero.

Invalidating entire ICache also clears the lockdown attribute for all cache lines. To invalidate part of the cache, use the Invalidate ICache Line instruction described below. Invalidating the entire ICache takes multiple cycles (Equal to the number of the tag entries).

Invalidate a single ICache line using MVA

Programmers can invalidate a single ICache line by using MVA through the following instruction:

```
MCR p15, 0, Rd, C7, C5, 1
```

The value in the Rd register is the line address that users want to invalidate. Its format is shown in Table 2-5.

Invalidating a single ICache line also clears the lockdown attribute of specific line.

Invalidate a single ICache line using set/index

Programmers can invalidate a single ICache line by using set/index through the following instruction:

```
        MCR p15, 0, Rd, C7, C5, 2
```

The value in the Rd register indicates that users want to invalidate a specific cache line. The format is described below:

RD[31:30]: This indicates the method to be invalidated.

RD[S+7:5]: This indicates the entry to be invalidated.

S: The logarithm to base-2 of the cache size in unit of Kbytes

Invalidating a single ICache line also clears its lockdown attribute.

Pre-fetch and Lockdown an ICache line

Programmers can lock down certain performance-critical code into ICache to prevent the code from being replaced during a program execution. The lockdown feature of ICache can be performed through programming the CP15 control register 9 by using the “Pre-fetch ICache Entry” instruction. Programmers must pay attention that the address of the lockdown entry should not overlap with the scratchpad.

Bit 31 of the CR9 register is used to invoke the lockdown operation. Writing ‘1’ to this bit enables the lockdown function. Writing ‘0’ to this bit disables the lockdown function. To lock down certain ICache lines in ICache, programmers must enable the lockdown function by using the following system coprocessor instruction.

```
        MCR p15, 0, Rd, C9, C0, 1
```

Then, use the following pre-fetch ICache line instruction to pre-fetch a desired line into ICache.

```
        MCR p15, 0, Rd, C7, C13, 1
```

The value in the Rd register is the line address to be pre-fetched.

A simple code fragment demonstrating the usage of ICache lockdown function is shown below:

```
MOV     r1, #0x30000000         ; r1=line address of certain line
MOV     r2, #0x80000000         ; r2=0x80000000
MCR     p15, 0, r2, c9, c0, 1   ; Writing 0x80000000 into CR9, enabling ICache lockdown function
MCR       p15, 0, r1, c7, c13, 1; Force line fetch from external memory
ADD       r1, r1, #16; Add 4-word offset to address
MCR       p15, 0, r1, c7, c13, 1; Force line fetch from external memory
……..
BIC       r0, r2, #0x80000000
MCR       p15, 0, r0, c9, c0, 1; Writing 0x0 into CR9, disabling ICache lockdown operation
```

In this example, two lines are pre-fetched from the external memory and stored and locked in ICache.

4.2 Data Cache (DCache)

The FA626TE DCache supports both accesses in the Write-Back (WB) and Write-Through (WT) modes. These modes are controlled by the cacheable and bufferable attributes defined in the page table, based on programming through the CP15 control registers 2. If a page or section has the attributes set as cacheable and bufferable, the accesses within this page or section will be in the Write-Back mode. If the page or section attributes are set to be cacheable but not bufferable, the accesses will be in the Write-Through mode. Please refer to Section 4.3 “Write Buffer” for further information. In FA626TE data cache, each cache line contains two dirty bits. Each dirty bit can record the status of half cache-line or sub-line (Four 32-bit words). When a cache replacement occurs, only the dirty sub-line should be written back to the external memory.

Write-Back mode

When a data write hits DCache, the corresponding cache half-line (Four words) will be updated and marked as “dirty” so that data will not be written to the external memory in the Write-Back mode. When a data write is missed in DCache, the corresponding line will be filled in the cache from an external memory so that data will be directly stored to the cache and not written to the external memory.

When a read or write is missed, a cache line may need to be evicted from DCache to leave a vacancy for the newly fetched line. The evicted sub-lines need to be written out to an external memory if it is marked as “dirty.”

Write-Through mode

When a data write hits the DCache, the cache line will be updated, data will be written to the external memory, and the cache line will not be marked as “dirty” in the Write-Through mode since data in the cache and the external memory are consistent. When a data write is missed in DCache, data will be directly written to the external memory. No cache line will be filled.

As described previously, the FA626TE data cache features the hit-under-miss access. The feature, “Hit under Miss”, means that the program execution will not be blocked even if a data has been missing in the cache, as long as the following instructions are not data dependent with the missing data and induce no further cache miss. The “Hit under Miss” feature can be disabled.

DCache is disabled by default. Programmers need to set bit 2 of the CR1 register in CP15 to enable DCache. Before or when DCache is enabled, MMU must be enabled by programming the CP15 register 1. On reset, the entire DCache content will be invalidated automatically.

FA626TE provides the following management operations for DCache:

1. Invalidate the entire DCache
2. Invalidate a single DCache line by using MVA
3. Invalidate a single DCache line by using set/index
4. Clean the entire DCache
5. Clean a single DCache line by using MVA
6. Clean a single DCache line by using set/index
7. Clean and invalidate the entire DCache
8. Clean and invalidate a single DCache line by using MVA
9. Clean and invalidate a single DCache line by using set/index
10. Pre-fetch and lockdown the DCache line

FA626TE does not provide a specific data pre-fetch instruction to lock down the data cache line. Instead, the “LDR” instruction is used to perform the data pre-fetch and lockdown if the lock bit in CR9 is set.

Invalidate entire DCache

Programmers can invalidate the entire DCache through the following system coprocessor instruction:

```
MCR    p15, 0, Rd, C7, C6, 0
```

The value in the Rd register should be zero.

This instruction invalidates the cache content without writing back the dirty lines to an external memory. Invalidating the entire DCache also clears the lockdown attributes of all cache lines. To invalidate part of the cache, use the "invalidate a single DCache line" instruction described in the next paragraph.

Invalidate a single DCache line using MVA

Programmers can invalidate a single DCache line by MVA through the following system coprocessor instruction:

```
MCR     p15, 0, Rd, C7, C6, 1
```

The value in the Rd register is the line address that users want to invalidate. The format of the Rd register is shown in Table 2-5.

Invalidating a single DCache line will directly invalidate a line without writing back the dirty line whether or not the lockdown attribute has been set on this line.

Invalidate a single DCache line using set/index

Programmers can invalidate a single DCache line by using set/index through the following system coprocessor instruction:

```
MCR     p15, 0, Rd, C7, C6, 2
```

The value in the Rd register is the set/index of DCache to be invalidated. The format of the Rd register is shown in Table 2-5.

Invalidating a single DCache line will directly invalidate the address line without writing back the line even if the line is dirty, and no matter the lockdown attribute has been set on this line or not.

Clean entire DCache

Programmers can clean the entire DCache by using the following system coprocessor instruction:

```
MCR    p15, 0, Rd, C7, C10, 0
```

The value in the Rd register should be zero. When this instruction is executed, all the “dirty” lines in DCache will be written back to the external memory and all cache lines will remain in DCache with cleared dirty bits. This operation synchronizes DCache with an external memory to ensure data consistence. To clean part of the cache, use the Clean a Single DCache Line instruction described in the next paragraph.

Clean a single DCache line using MVA

Programmers can clean a single DCache line using MVA with the following system coprocessor instruction:

```
MCR    p15, 0, Rd, C7, C10, 1
```

The value in the Rd register is the line address that users want to invalidate. When this instruction is executed, the addressed line in DCache will be written back to the external memory if it is marked as “dirty.” The line will remain in DCache. This operation synchronizes DCache with an external memory to enforce the program order.

Clean a single DCache line using set/index

Programmers can clean a single DCache line using set/index with the following system coprocessor instruction:

```
MCR    p15, 0, Rd, C7, C10, 2
```

The value in the register Rd is the set/index of the DCache that users want to invalidate. When this instruction is executed, the addressed line in DCache will be written back to the external memory if it is marked as “dirty.” The line will remain in DCache. This operation synchronizes DCache with external memory to enforce the program order.

Clean and invalidate entire DCache

Programmers can clean and invalidate the entire DCache by using the following instruction:

```
MCR    p15, 0, Rd, C7, C14, 0
```

The value in the Rd register should be zero. When this instruction is executed, all “dirty” lines in DCache will be written back to the external memory and all cache lines will be invalidated. To clean and invalidate part of the cache, programmers should use the “clean and invalidate a single DCache line” instruction described in the next paragraph. It takes the number of b * D + #lines * 2 cycles. “D” indicates the number of dirty sub-lines and “b” indicates the number of bus cycles for writing out each sub-line.

Clean and invalidate a single DCache line using MVA

Programmers can clean and invalidate a single DCache line using MVA with the following instruction:

```
MCR    15, 0, Rd, C7, C14, 1
```

The value of the Rd register should be the line address that users want to clean and invalidate. When this instruction is executed, the addressed line in DCache will be written back to the external memory if it is marked as “dirty” and the address line will be invalidated.

Clean and invalidate a single DCache line using set/index

Programmers can clean and invalidate a single DCache line using set/index with the following instruction:

```
MCR    15, 0, Rd, C7, C14, 2
```

The value of the Rd register is set/index of DCache that users want to clean and invalidate. When this instruction is executed, the address line in DCache will be written back to the external memory if it is marked as “dirty” and the line will be invalidated.

Pre-fetch and lockdown a DCache line

Programmers can lock down certain performance-critical data in DCache to prevent the data from being evicted from cache during the program execution. The lockdown function of DCache can be performed by programming the CP15 control register 9 and by using the LDR instruction. The lines in DCache can be unlocked through the invalidated DCache line operation.

Bit 31 of the CR9 register is used to invoke the cache lockdown operation. Writing ‘1’ to this bit will enable the lockdown operation. Writing ‘0’ to this bit will disable the lockdown operation. To lock down the DCache lines in DCache, programmers must enable the lockdown function by using the following instruction:

```
MCR    p15, 0, Rd, C9, C0, 0
```

Then the LDR instruction should be used to pre-fetch the required line to DCache. A simple code fragment demonstrating how to use the DCache lockdown function is shown below:

```
MOV      r1, 0x30000000           ; r1 = Line address of a certain line
MOV      r2, #0x80000000          ; r2 = 0x80000000
MCR      p15, 0, r2, c9, c0, 0    ; Writing 0x80000000 into CR9, enabling the DCache lockdown
                                    function
LDR      r3, [r1]+16              ; Force line fetch from the external memory
LDR      r3, [r1]                 ; Force line fetch from the external memory
……..
BIC                               r0, r2, #0x80000000
MCR      p15, 0, r0, c9, c0, 0    ; Writing 0x0 into CR9, disabling the DCache lockdown operation
```

In this example, two lines are pre-fetched from the external memory and stored and locked in DCache. These two locked lines can still be read or written.

4.3 Write Buffer

FA626TE uses a write buffer to enhance the overall performance of the processor. With the write buffer, the processor core does not have to wait for the completion of a write on the bus. Instead, the write is putted into the write buffer and the processor will continue executing the instructions. Data in the write buffer will be written back to the external memory whenever the bus is available. The write buffer snooping is provided to ensure the correctness of a program execution. When a read hits data in the write buffer, the read operation will be pending until the hit data in the write buffer is written out.

The write buffer is disabled in default. However, it will be automatically enabled when MMU is turned on. Users can still disable (Or re-enable) the write buffer after the MMU is turned on. However, the write buffer should be drained before being disabled.

The cacheability and bufferability attributes of a page or section determine the data caching and buffering behaviors. Table 4-1 shows four types of access modes. Each access mode is described below.

Table 4-1. Caching and Buffering Behaviors

Cacheability (With DCache and MMU Enabled)	Bufferability (With MMU Enabled)	Access Mode
Non-cacheable	Non-bufferable	NCNB
Non-cacheable	Bufferable	NCB
Cacheable	Non-bufferable	WT (Write-Through)
Cacheable	Bufferable	WB (Write-Back)

Four types of access modes:

- **Non-Cacheable Non-Bufferable (NCNB)**

  Data are not cached in DCache and are not buffered in the write buffer.

  Both read and write accesses are directed to the external bus. The write buffer is drained automatically before the NCNB access. CPU must wait for the access completion on the external bus to execute the next instruction. Both read and write accesses can be aborted by the external devices.

- **Non-Cacheable but Bufferable (NCB)**

  Data are not cached in DCache, but is buffered in the write buffer.

  The read accesses are directly reverted to the external memory. The write data are located in the write buffer and passed to the external memory whenever the bus access is available. For read access, the write buffer is drained automatically before a read access.

  The operating of aborting a read access is passed to CPU; while the operating of aborting a write access is ignored. If an SWP instruction operates on the NCB data, it will be regarded as an NCNB access.

- **Write-Through (WT) access**

  If a read access hits DCache, the read data will come from DCache. Otherwise, the read access will be passed to the external memory and a cache line will be fetched and stored in DCache.

  If a write access hits DCache, the write data will be updated to DCache and the external memory. The cache line will not be marked as “dirty”. The write data will be located in the write buffer and passed to the external memory whenever the bus is available. If a write is missing in DCache, the write data will be located in the write buffer. No cache line fill will be performed.

  Both read and write requests cannot be aborted by the external memory. FA626TE will ignore the externally generated data abort for this type of access.

- **Write-Back (WB) access**

  If a read address hits DCache, the read data will come from DCache. Otherwise, the read access will be passed to the external memory and a cache line will be fetched and stored in DCache.

  If a write hits DCache, the write data will be stored to DCache and the modified cache line will be marked as “dirty”. If a write is missing in DCache, the cache line will be fetched and stored in DCache. The “dirty” cache lines will be written back to the external memory when these lines are replaced or when a clean DCache instruction is executed.

  The external memory cannot abort both read and write accesses at a time. FA626TE will ignore the externally generated abort for this type of access.

  The FA626TE write buffer has eight address entries and eight double-word data entries. A dirty half-line being replaced occupies only one address entry and two data entries. In all other cases, each write occupies one address entry and one data entry no matter what the access size is.

Chapter 5

DMA (AXI Interface Only)

This chapter contains the following sections:

5.1 DMA Function Mechanism

- Several registers in CP15 is for programming the DMA control.
- Two independent channels for external memory and DSPAD. Data should be transferred one at a time. If one channel attempts to transfer when the other is transferring, it is queued.
- Each channel has four states: Idle, running, queued, and complete (Containing error) states.
- Some registers can only be accessed in the privileged mode, while the other can access in both the privileged or user mode by programming.
- DMA is programmed by using the virtual addresses.

5.2 DMA Identification and Status Registers

The purposes of the DMA identification and status registers are sued to define:

- The DMA channels that are physically implemented on FA626.
- The current status of the DMA channels

Processes that handle DMA can read this register to determine the physical resources implemented and the availability.

The command format is:
MRC p15, 0, <Rd>, c11, c0, opcode_2

The following table shows the format of the DMA identification and status registers 0-3.

31	2	1	0
UNP		CH1	CH0

The following table shows the bit function of the DMA identification and status registers.

Bit Range	Field Name	Function
[31:2]	-	UNP/SBZ
1	CH1	This bit provides the information of the DMA channel 1 function. 0 = Disable the DMA channel 1 function 1 = Enable the DMA channel 1 function
0	CH0	This bit provides the information of the DMA channel 0 function. 0 = Disable the DMA channel 0 function 1 = DMA Channel 0 function enabled.

The following table shows the opcode_2 function of the DMA identification and status registers.

Opcode_2	Function
0	This bit indicates the present channel. 0 = The channel is not present. 1 = The channel is present.
1	This bit indicates queued channel. 0 = The channel is not queued. 1 = The channel is queued.
2	This bit indicates the running channel running 0 = The channel is not running. 1 = The channel is running.
3	This bit indicates the interrupting channel 0 = The channel is not Interrupting. 1 = The channel is Interrupting through the completion or an error.

Example codes:

MRC p15, 0, <Rd>, c11, c0, 0; Read the DMA Identification and Status Register of "present"

MRC p15, 0, <Rd>, c11, c0, 1; Read the DMA Identification and Status Register of "queued"

MRC p15, 0, <Rd>, c11, c0, 2; Read the DMA Identification and Status Register of "running"

MRC p15, 0, <Rd>, c11, c0, 3; Read the DMA Identification and Status Register of "interrupting"

The identification and status registers can only be read in the privileged mode.

5.3 DMA User Accessibility Register

The purpose of the DMA User Accessibility Register is to determine if a User mode process can access the registers for each channel.

The following table lists the format of the DMA user accessibility register.

31	2	1	0
UNP/SBZ		U1	U0

The following table shows the bit function of the DMA user accessibility register.

Bit Range	Field Name	Function
[31:2]	-	UNP/SBZ
1	U1	This bit indicates if a user mode process can access the registers for channel 1. 0 = User mode cannot access channel 1 (Reset value). In this case, the user-mode accesses will cause an Undefined exception. 1 = User mode can access channel 1.
0	U0	This bit indicates if a user mode process can access the registers for channel 0. 0 = User mode cannot access channel 0 (Reset value). In this case, the user-mode accesses will cause an Undefined exception. 1 = User mode can access channel 0.

Example codes:

MRC p15, 0, <Rd>, c11, c1, 0; Read from the DMA user accessibility register

MCR p15, 0, <Rd>, c11, c1, 0; Write to the DMA user accessibility register

The registers listed below can be accessed in the user mode when the U bit = ‘1’ for the current channel.

- DMA enable registers
- DMA control register
- DMA internal start address register
- DMA external start address register
- DMA internal end address register
- DMA channel status register on page

Users can access the DMA channel number registers in the user mode when the U bit for any channel is set to ‘1’. The contents of these registers must be preserved on a task switch if the registers are accessible by user. The DMA user accessibility register can only be accessed in the privileged mode.

5.4 DMA Channel Number Register

The purpose of the DMA channel number register is to select a DMA channel.

The following table shows the format of the DMA channel number register

31	1	0
UNP/SBZ		CN

The following table shows the bit function of the DMA channel number register.

Bit Range	**Field Name**	**Function**
[31:1]	-	UNP/SBZ
0	CN	This bit indicates the selected DMA channel. 0 = Select the DMA channel 0 (Reset value) 1 = Select the DMA channel 1

Example codes:

MRC p15, 0, <Rd>, c11, c2, 0; Read from the DMA Channel Number Register

MCR p15, 0, <Rd>, c11, c2, 0; Write to the DMA Channel Number Register

The processor can access this register in the user mode if the U bit of c11, the DMA user accessibility register, for any channel is set to '1'

5.5 DMA Enable Registers

The purpose of the DMA enable registers is to start, stop, or clear the DMA transfers for each implemented channel.

The following table shows the commands that operate through the registers.

Stop	The DMA channel stops performing the memory accesses right after the level-one DMA issues an instruction. DMA can issue a "Stop" command when the channel status is Running. The DMA channel will take several cycles to stop after DMA issues a Stop instruction. The channel status remains at Running until the DMA channel stops. The channel status is set to Complete or Error at the point that all outstanding memory accesses are complete. The start address registers contain the addresses that DMA requires to restart the operation when the channel stops. If the "Stop" command occurs when the channel status is Queued, the channel status will change to Idle. The "Stop" command has no effect if the channel status is not Running or Queued.
Start	The "Start" command causes the channel to start the DMA transfers. If other DMA channel is not in operation, the channel status will be set to Running on the execution of a Start command. If other DMA channel is operational, the channel status will be set to Queued. A channel is operational if either: • The channel status is Queued. • The channel status is Running. • The channel status will be Complete or Error when the internal or external address error status register indicates an Error.
Clear	The "Clear" command changes the channel status from Complete or Error to Idle. It also clears: • All Error bits for the DMA channel The interrupt set by the DMA channel will result as an error or completion, The "Clear" command does not change the contents of the internal and external start address registers. A "Clear" command has no effect when the channel status is Running or Queued.

Debug implications for DMA

The level-1 DMA behaves as a separate engine from the processor core; when started, it will work autonomously. When the level-1 DMA has channels with the status of Running or Queued, these channels will continue to run or start running, even if a debug mechanism stops the processor. This will change the contents of the scratchpad while the processor stops debugging. To avoid this situation, users must ensure that the level-1 DMA issues a “Stop” command to stop the Running or Queued channels when entering the debug mode.

Example codes:

MCR p15, 0, <Rd>, c11, c3, 0; Stop the DMA enable register

MCR p15, 0, <Rd>, c11, c3, 1; Start the DMA enable register

MCR p15, 0, <Rd>, c11, c3, 2; Clear the DMA enable register

The processor can write this register in the user mode if the U bit of c11, the DMA user accessibility register, for any channel is set to ‘1’. They are write-only registers.

5.6 DMA Control Register

The purpose of the DMA control register for each channel is to control the operations of that DMA channel.

The following table shows the DMA control register

31	30	29	28	27	26	25 20	19 8	7 2	1 0
TR	DT	IC	IE	FT	UM	UNP/SBZ	ST	UNP/SBZ	TS

The following table shows the bit function of the DMA control register.

Bit Range	Field Name	Function
31	TR	This bit indicates the target scratchpad. 0 = Data scratch pad (Reset value) 1 = Instruction scratchpad (Not support)

Bit Range	Field Name	Function
30	DT	This bit indicates the direction of a transfer. 0 = Transfer from the level-2 memory to the scratchpad (Reset value) 1 = Transfer from the scratchpad to the level-2 memory
29	IC	This bit indicates whether the DMA channel must assert an interrupt on completion of the DMA transfer, or if the DMA is stopped by a “Stop” command. The interrupt is de-asserted, from this source, if the processor performs a Clear operation on the channel that caused the interrupt. The U bit has no effect on whether an interrupt is generated on completion: 0 = No interrupt on Completion (Reset value) 1 = Interrupt on Completion (Please refer to c11, the DMA user accessibility register.)
28	IE	This bit indicates that the DMA channel must assert an interrupt on an error. The interrupt will be de-asserted from this source when the channel is set to Idle with a Clear operation. 0 = No Interrupt on Error, if the U bit is ‘0’ (Reset value) 1 = Interrupt on Error, regardless of the U bit. All DMA transactions on the channels that have the U bit set to ‘1’ Interrupt on Error regardless of the value written into this bit. (Please refer to c11, the DMA user accessibility register.)
27	FT	Ignore write when this bit is read as ‘1’ In FA626, this bit has no effect.
26	UM	This bit indicates that the permission checks based on DMA are in the user or privileged mode. The UM bit is provided so that the User mode can be emulated by a privileged mode process. For a User mode process the setting of the UM bit is irrelevant and behaves as if set to 1: 0 = Transfer is a privileged transfer (Reset value). 1 = Transfer is a user-mode transfer.
[25:20]	-	UNP/SBZ
[19:8]	ST	This field indicates the increment on the external address between each consecutive access of the DMA. A Stride of zero, reset value, indicates that the external address is not to be incremented. This is designed to facilitate the accessing of the volatile locations, such as a FIFO. The Stride is interpreted as a positive number or zero. The internal address increment is not affected by the Stride, but the transaction size is fixed. The value of Stride is in bytes. The value of Stride must be aligned to the transaction size; otherwise, this will result in a bad parameter error.
[7:2]	-	UNP/SBZ

[1:0]	TS	This field indicates the size of the transactions that the DMA channel performs.
		b00 = Byte (Reset value)
		b01 = Half-word
		b10 = Word
		b11 = Double-word, 8 bytes

Example codes:

MRC p15, 0, <Rd>, c11, c4, 0; Read from the DMA control register

MCR p15, 0, <Rd>, c11, c4, 0; Write to the DMA control register

When the channel has the status of Running or Queued, any attempt to write to the DMA control register will be ignored.

The processor can access this register in the user mode if the U bit of c11, the DMA user accessibility register, for any channel is set to '1'.

5.7 DMA Internal Start Address Register

The purpose of the DMA internal start address register for each channel is to define the first address in the scratchpad for that channel. That is, it defines the first address that a data transfer goes to or comes from.

The DMA internal start address register bits [31:0] contain the internal start VA.

Example codes:

MRC p15, 0, <Rd>, c11, c5, 0; Read from the DMA internal start address register

MCR p15, 0, <Rd>, c11, c5, 0; Write to the DMA internal start address register

The internal start address is a VA. The page tables describe the physical mapping of VA when the channel starts. It must use the same translation tables as the internal end address. It means that they must have the same virtual-to-physical address mapping, memory access permissions, and cachability and bufferability bits.

The memory attributes for that VA are used in the transfer so that the memory permission faults may be generated. The internal start address must lie with a scratchpad; otherwise, an error will be reported in the DMA channel status register.

The contents of this register will not be changed when the DMA channel has a status of Running. When the channel is stopped because of a Stop command or an error, it contains the address required to restart the transaction. On completion, it contains the address that equals to the internal end address. When an IS error occurs, it keeps its original value. When an ES error occurs, it contains the next address not accessed.

The internal start address must be aligned to the transaction size set in the DMA control register or the processor that generates a bad parameter error.

When the channel has the status of Running or Queued, any attempt to write to the DMA internal start address register will be ignored.

The processor can access this register in the user mode if the U bit of c11, the DMA user accessibility register, for any channel is set to '1'.

5.8 DMA External Start Address Register

The purpose of the DMA external start address register for each channel is used to define the first address in the external memory for that DMA channel. That is, it defines the first address that a data transfer goes to or comes from.

The DMA external start address register bits [31:0] contain the external start VA.

Example codes:

MRC p15, 0, <Rd>, c11, c6, 0; Read from the DMA external start address register

MCR p15, 0, <Rd>, c11, c6, 0; Write to the DMA external start address register

The external start address is a VA, the physical mapping that must be described in the page tables at the time that the channel is started. The memory attributes for that VA are used in the transfer so that the memory permission faults may be generated.

The external start address must lie in the external memory outside the level-1 memory system; otherwise, the results are Unpredictable. The global system behavior, not including the security, can be affected.

This register contents do not change while the DMA channel status is Running. When the channel stops because of a “Stop” command, or an error, it contains the address that the DMA requires to restart the transaction. On completion, it contains the address that equals to the final address of the transfer accessed plus the Stride. When an IS error occurs, it keep its original value. When an ES error occurs but not an external abort, it contains the address that causes this error. When an ES error occurs due to an external abort, it contains the next address not accessed if DMA performs AXI single transfer or the next burst address not accessed if DMA performs an AXI burst transfer.

If the external start address does not align with the transaction size that is set in the control register, the processor will generate a bad parameter error.

While the channel has the status of Running or Queued, any attempt of writing to the DMA external start address register will be ignored.

The processor can access this register in the user mode if the U bit of c11, the DMA user accessibility register, for any channel is set to ‘1’.

5.9 DMA Internal End Address Register

The purpose of the DMA internal end address register for each channel is used to define the final internal address for that channel, which is the end address of a data transfer.

The DMA Internal End Address Register bits [31:0] contain the Internal End VA.

Example codes:

```
MRC p15, 0, <Rd>, c11, c7, 0; Read from the DMA internal end address register
MCR p15, 0, <Rd>, c11, c7, 0; Write to the DMA internal end address register
```

The internal end address is the final internal address modulo the transaction size that DMA will access plus the transaction size; therefore, the internal end address is the first, incremented, address that DMA does not access.

If the internal end address is the same of the internal start address, the DMA transfer will complete immediately without performing any transaction.

When the transaction associated with the final internal address has completed, the whole DMA transfer is complete.

The internal end address is a VA. The page tables describe the physical mapping of VA when the channel starts. It must use the same translation tables as the internal start address. It means that they must have the same virtual-to-physical address mapping, memory access permissions, and cachability and bufferability bits.

The memory attributes for that VA are used in the transfer so that the memory permission faults may be generated. The internal end address must lie with a scratchpad; otherwise, an error will be reported in the DMA channel status register.

The internal end address must be aligned to the transaction size set in the DMA control register or the processor generated a bad parameter error.

When the channel has the status of Running or Queued, any attempt to write to the DMA internal end address register will be ignored.

The processor can access this register in the user mode if the U bit of c11, the DMA user accessibility register, for any channel is set to ‘1’.

5.10 DMA Channel Status Register

The purpose of the DMA channel status register for each channel is used to define the status of the most recently started DMA operation on that channel.

The following table shows the format of the DMA channel status register.

31 17	16	15 14	13	12	11 7	6 2	1 0
UNP/SBZ	ESX[0]	UNP/SBZ	ISX[0]	BP	ES	IS	Status

The following table shows the bit function of the DMA channel status register.

Bit Range	Field Name	Function
[31:17]	-	UNP/SBZ
16	ESX[0]	The ESX[0] bit adds a SLVERR or DECERR qualifier to the ES encoding. It is only predictable on the ES encodings of b11010; otherwise, UNP/SBZ. For the predictable encodings: 0 = DECERR 1 = SLVERR.
[15:14]	UNP/SBZ	-
13	ISX[0]	-
12	BP	This bit indicates the status of the DMA parameters. 0 = DMA parameters are acceptable (Reset value). 1 = DMA parameters are conditioned inappropriately. DMA checks the following case after a Start command: • The internal start, external start, and external end addresses and the Stride must be multiples of the transaction size. • The internal start address is smaller than or equals to the internal end address. • If the DMA AXI bus is configured as 32-bit, the TS bits of the DMA control register can only accept 2'b00, 2'b01, or 2'b10. If this is not one of these cases, the BP bit should be set to '1' and the DMA channel should not be started and stays at the Idle status.
[11:7]	ES	This field indicates the status of the external address error. All other encodings are reserved. b00000 = No error (Reset value) b11010 = External abort can be imprecise. b11100 = External Abort on translation of the page table or page table data abort Others are reserved.

Bit Range	Field Name	Function
[6:2]	IS	This field indicates the status of the internal address error. All other encodings are reserved. b00000 = No error (Reset value) b01000 = Scratchpad out of range b11100 = External abort on translation of page table or page table data abort Others are reserved.

[1:0]	Status	This field indicates the status of the DMA channel.
		b00 = Idle (Reset value)
		b01 = Queued
		b10 = Running
		b11 = Complete or Error

Example code:

MRC p15, 0, <Rd>, c11, c8, 0; Read from the DMA channel status register

In the event of an error, the appropriate start address register contains the faulted address, unless the error is an external error that is set to b11010 by using bits [11:7].

A channel with the state of Queued will automatically change to Running if other channel, if implemented, changes to Idle, Complete, or Error, has no error.

When a channel completes all transfers of DMA that will change to the memory locations caused by those transfers that are visible to other observers; its status is changed from Running to Complete or Error. This change does not happen before the external accesses from the transfer complete.

If the channel is in the event of an error the processor asserts the DMAIRQN pin provided. In the event of an external abort on a page table walk caused by DMA, the processor asserts the DMAIRQN output.

The processor can read this register in User mode if the U bit of c11, the DMA user accessibility register, for any channel is set to '1'. They are read only.

Chapter 6

AHB Bus Interface Unit

This chapter contains the following sections:

6.1 AHB Control Signals

6.1.1 HTRANS[1:0]

6.1.1.1 I Master Port

IR64 = 1

IRHTRANS[1:0]	Transfer Type
b00	IDLE
b10	NSEQ
b11	SEQ

IR64 = 0

IRHTRANS[1:0]	Transfer Type
b00	IDLE
b10	NSEQ
b11	SEQ

6.1.1.2 DR Master Port

DR64 = 1

DRHTRANS[1:0]	Transfer Type
b00	IDLE
b10	NSEQ
b11	SEQ

DR64 = 0

DRHTRANS[1:0]	Transfer Type
b00	IDLE
b10	NSEQ
b11	SEQ

6.1.1.3 DW Master Port

DW64 = 1

DWHTRANS[1:0]	Transfer Type
b00	IDLE
b10	NSEQ
b11	SEQ

DW64 = 0

DWHTRANS[1:0]	Transfer Type
b00	IDLE
b10	NSEQ
b11	SEQ

6.1.1.4 P Master Port

PHTRANS[1:0]	Transfer Type
b00	IDLE
b10	NSEQ
b11	SEQ

6.1.2 HSIZE[2:0]

6.1.2.1 I Master Port

IR64 = 1

IRHSIZE[2:0]	Byte in Transfer
b011	8

IR64 = 0

IRHSIZE[2:0]	Byte in Transfer
b010	4

6.1.2.2 DR Master Port

DR64 = 1

DRHSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4
b011	8

DR64 = 0

DRHSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4

6.1.2.3 DW Master Port

DW64 = 1

DWHSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4
b011	8

DW64 = 0

DWHSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4

6.1.2.4 P Master Port

PHSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4

6.1.3 HBURST[2:0]

6.1.3.1 I Master Port

IR64 = 1

IRHBURST[2:0]	Burst Type
b000	SINGLE
b010	WRAP4

IR64 = 0

IRHBURST[2:0]	Burst Type
b000	SINGLE
b100	WRAP8

6.1.3.2 DR Master Port

DR64 = 1

DRHBURST[2:0]	Burst Type
b000	SINGLE
b010	WRAP4

DR64 = 0

DRHBURST[2:0]	Burst Type
b000	SINGLE
b100	WRAP8

6.1.3.3 DW Master Port

DW64 = 1

DWHBURST[2:0]	Burst Type
b000	SINGLE
b001	INCR
b011	INCR4

DW64 = 0

DWHBURST[2:0]	Burst Type
b000	SINGLE
b001	INCR
b011	INCR4
b101	INCR8

6.1.3.4 P Master Port

PHBURST[2:0]	Burst Type
b000	SINGLE

6.1.4 HLOCK

6.1.4.1 I Master Port

IRHLOCK	Description
b0	Normal access

6.1.4.2 DR Master Port

DRHLOCK	Description
b0	Normal access
b1	Locked access

6.1.4.3 DW Master Port

DWHLOCK	Description
b0	Normal access

6.1.4.4 P Master Port

PHLOCK	Description
b0	Normal access

6.1.5 HPROT[3:0]

6.1.5.1 I Master Port

IRHPROT[3:2]	AHB Transaction Attribute	Memory Attribute C or B
b00	NCNB	NCNB
b01	NCB	NCB
b10	Write-Through	CNB
b11	Write-Back	CB

IRHPROT[1:0]	Description
IRHPROT[1]	0 = User 1 = Privileged
IRHPROT[0]	0 = Instruction access

6.1.5.2 DR Master Port

DRHPROT[3:2]	AHB Transaction Attribute	Memory Attribute C or B
b00	NCNB	NCNB
b01	NCB	NCB
b10	Write-Through	CNB
b11	Write-Back	CB

DRHPROT[1:0]	Description
DRHPROT[1]	0 = User 1 = Privileged
DRHPROT[0]	0 = Data access

6.1.5.3 DW Master Port

DWHPROT[3:2]	AHB Transaction Attribute	Memory Attribute C or B	CR1-0 RAO Bit	CR1-0 STMWA Bit	STR/STM Cache Miss
b00	NCNB	NCNB	-	-	-
b01	NCB	NCB	-	-	-
b10	Write-Through	CNB	-	-	-
b10	Write-Through	CB	1	-	-
b11	Write-Back	CB	0	1	-
b10	Write-Through	CB	0	0	STM cache miss
b11	Write-Back	CB	0	0	STR cache miss

DWHPROT[1:0]	Description
DWHPROT[1]	0 = User 1 = Privileged
DWHPROT[0]	1 = Data access

6.1.5.4 P Master Port

PHPROT[3:2]	AHB Transaction Attribute	Memory Attribute C or B
b00	NCNB	NCNB
b01	NCB	NCB

PHPROT[1:0]	Description
PHPROT[1]	0 = User 1 = Privileged
PHPROT[0]	1 = Data access

6.2 AHB Addressing Signals

6.2.1 I Master Port

6.2.1.1 Cacheable Fetch

IR64 = 1

CPU R/W Address[4:0]	First IRHADDR[4:0]	IRHBURST	IRHSIZE
0x00	0x00	WRAP4	64-bit
0x04	0x00	WRAP4	64-bit
0x08	0x08	WRAP4	64-bit
0x0C	0x08	WRAP4	64-bit
0x10	0x10	WRAP4	64-bit
0x14	0x10	WRAP4	64-bit
0x18	0x18	WRAP4	64-bit
0x1C	0x18	WRAP4	64-bit

IR64 = 0

CPU R/W Address[4:0]	First IRHADDR[4:0]	IRHBURST	IRHSIZE
0x00	0x00	WRAP8	32-bit
0x04	0x00	WRAP8	32-bit
0x08	0x08	WRAP8	32-bit
0x0C	0x08	WRAP8	32-bit
0x10	0x10	WRAP8	32-bit
0x14	0x10	WRAP8	32-bit
0x18	0x18	WRAP8	32-bit
0x1C	0x18	WRAP8	32-bit

If the SPLIT/RETRY/ERROR response or the early-burst termination occurs, users should set IRHTRANS = NSEQ and IRHBURST = SINGLE for the remaining transfers.

6.2.1.2 Non-cacheable Fetch

IR64 = 1

CPU R/W Address[4:0]	First IRHADDR[4:0]	IRHBURST	IRHSIZE
0x00	0x00	SINGLE	64-bit
0x04	0x00	SINGLE	64-bit
0x08	0x08	SINGLE	64-bit
0x0C	0x08	SINGLE	64-bit
0x10	0x10	SINGLE	64-bit
0x14	0x10	SINGLE	64-bit
0x18	0x18	SINGLE	64-bit
0x1C	0x18	SINGLE	64-bit

IR64 = 0

CPU R/W Address[4:0]	First IRHADDR[4:0]	IRHBURST	IRHSIZE
0x00	0x00	SINGLE	32-bit
0x04	0x00	SINGLE	32-bit
0x08	0x08	SINGLE	32-bit
0x0C	0x08	SINGLE	32-bit
0x10	0x10	SINGLE	32-bit
0x14	0x10	SINGLE	32-bit
0x18	0x18	SINGLE	32-bit
0x1C	0x18	SINGLE	32-bit

6.2.2 DR Master Port

6.2.2.1 Line Fill

DR64 = 1

CPU R/W Address[4:0]	First DRHADDR[4:0]	DRHBURST	DRHSIZE
0x00 ~ 0x07	0x00	WRAP4	64-bit
0x08 ~ 0x0F	0x08	WRAP4	64-bit
0x10 ~ 0x17	0x10	WRAP4	64-bit
0x18 ~ 0x1F	0x18	WRAP4	64-bit

DR64 = 0

CPU R/W Address[4:0]	First DRHADDR[4:0]	DRHBURST	DRHSIZE
0x00 ~ 0x03	0x00	WRAP8	32-bit
0x04 ~ 0x07	0x04	WRAP8	32-bit
0x08 ~ 0x0B	0x08	WRAP8	32-bit
0x0C ~ 0x0F	0x0C	WRAP8	32-bit
0x10 ~ 0x13	0x10	WRAP8	32-bit
0x14 ~ 0x17	0x14	WRAP8	32-bit
0x18 ~ 0x1B	0x18	WRAP8	32-bit
0x1C ~ 0x1F	0x1C	WRAP8	32-bit

If the SPLIT/RETRY/ERROR response or the early-burst termination occurs, users should set DRHTRANS = NSEQ and DRHBURST = SINGLE for the remaining transfers.

6.2.2.2 Non-cacheable LDRB

CPU R/W Address[2:0]	DRHADDR[2:0]	DRHBURST	DRHSIZE
0x0	0x0	SINGLE	8-bit
0x1	0x1	SINGLE	8-bit
0x2	0x2	SINGLE	8-bit
0x3	0x3	SINGLE	8-bit
0x4	0x4	SINGLE	8-bit
0x5	0x5	SINGLE	8-bit
0x6	0x6	SINGLE	8-bit
0x7	0x7	SINGLE	8-bit

6.2.2.3 Non-cacheable LDRH

CPU R/W Address[2:0]	DRHADDR[2:0]	DRHBURST	DRHSIZE
0x0	0x0	SINGLE	16-bit
0x2	0x2	SINGLE	16-bit
0x4	0x4	SINGLE	16-bit
0x6	0x6	SINGLE	16-bit

6.2.2.4 Non-cacheable LDR or LDM1

CPU R/W Address[2:0]	DRHADDR[2:0]	DRHBURST	DRHSIZE
0x0	0x0	SINGLE	32-bit
0x4	0x4	SINGLE	32-bit

6.2.2.5 Non-cacheable LDM2 ~ 16/LDRD

CPU R/W Address[2:0]	Operation
0x0	2 ~ 16 single transfers No burst transfer
0x4	2 ~ 16 single transfers No burst transfer

6.2.3 DW Master Port

6.2.3.1 Full line Write-back

DW64 = 1

CPU R/W Address[4:0]	DWHADDR[4:0]	DWHBURST	DWHSIZE
Evicted cache line valid and both halves dirty	0x00	INCR4	64-bit

DW64 = 0

CPU R/W Address[4:0]	DWHADDR[4:0]	DWHBURST	DRHSIZE
Evicted cache line valid and both halves dirty	0x00	INCR8	32-bit

If the SPLIT/RETRY/ERROR response or the early-burst termination occurs, users should set DWHTRANS = NSEQ and DWHBURST = INCR for the remaining transfers.

6.2.3.2 Half-line Write-back

DW64 = 1

CPU R/W Address[4:0]	DWHADDR[4:0]	DWHBURST	DWHSIZE
Evicted cache line valid and lower half dirty	0x00	INCR	64-bit
Evicted cache line valid and higher half dirty	0x10	INCR	64-bit

DW64 = 0

CPU R/W Address[4:0]	DWHADDR[4:0]	DWHBURST	DWHSIZE
Evicted cache line valid and lower half dirty	0x00	INCR4	32-bit
Evicted cache line valid and higher half dirty	0x10	INCR4	32-bit

If the SPLIT/RETRY/ERROR response or the early-burst termination occurs, users should set DWHTRANS = NSEQ and DWHBURST = INCR for the remaining transfers.

6.2.3.3 Cacheable Write-Through or Non-cacheable STRB

DW64 = 1

CPU R/W Address[2:0]	DWHADDR[2:0]	DWHBURST	DWHSIZE
0x0	0x0	INCR	8-bit
0x1	0x1	INCR	8-bit
0x2	0x2	INCR	8-bit
0x3	0x3	INCR	8-bit
0x4	0x4	INCR	8-bit
0x5	0x5	INCR	8-bit
0x6	0x6	INCR	8-bit
0x7	0x7	INCR	8-bit

DW64 = 0

CPU R/W Address[2:0]	DWHADDR[2:0]	DWHBURST	DWHSIZE
0x0	0x0	INCR	8-bit
0x1	0x1	INCR	8-bit
0x2	0x2	INCR	8-bit
0x3	0x3	INCR	8-bit
0x4	0x4	INCR	8-bit
0x5	0x5	INCR	8-bit
0x6	0x6	INCR	8-bit
0x7	0x7	INCR	8-bit

6.2.3.4 Cacheable Write-Through or Non-cacheable STRH

DW64 = 1

CPU R/W Address[2:0]	DWHADDR[2:0]	DWHBURST	DWHSIZE
0x0	0x0	INCR	16-bit
0x2	0x2	INCR	16-bit
0x4	0x4	INCR	16-bit
0x6	0x6	INCR	16-bit

DW64 = 0

CPU R/W Address[2:0]	DWHADDR[2:0]	DWHBURST	DWHSIZE
0x0	0x0	INCR	16-bit
0x2	0x2	INCR	16-bit
0x4	0x4	INCR	16-bit
0x6	0x6	INCR	16-bit

6.2.3.5 Cacheable Write-Through or Non-cacheable STR or STM1

DW64 = 1

CPU R/W Address[2:0]	DWHADDR[2:0]	DWHBURST	DWHSIZE
0x0	0x0	INCR	32-bit
0x4	0x4	INCR	32-bit

DW64 = 0

CPU R/W Address[2:0]	DWHADDR[2:0]	DWHBURST	DWHSIZE
0x0	0x0	INCR	32-bit
0x4	0x4	INCR	32-bit

6.2.3.6 Cacheable Write-Through or Non-cacheable STM2~16/STRD

Both DW64 = 0 or 1

CPU R/W Address[2:0]	Operation
0x0	If STM is in the NCNB region, it always issues multiple INCR writes with the length of 1; otherwise, STM is split into several INCR burst transfers depending on the available data in the write buffer. When the bus interface issues a write command, it merges the continuous data in the write buffer into a burst write.
0x4	

Users should set DWHTRANS = NSEQ/SEQ, DWHBURST = INCR, and DWHSIZE = 32-bit. DW64 = ‘0’ or ‘1’, DWHSIZE is fixed to 32-bit

6.2.3.7 Multiple Cacheable Write-Through or Non-cacheable STRB/STRH/STR/STRD with Continuous Address

It is the same cases as the cacheable Write-Through or Non-cacheable STM2 ~ 16/STRD.

6.2.4 P Master Port

6.2.4.1 Non-cacheable LDRB/STRB

CPU R/W Address[2:0]	PHADDR[2:0]	PHBURST	PHSIZE
0x0	0x0	SINGLE	8-bit
0x1	0x1	SINGLE	8-bit
0x2	0x2	SINGLE	8-bit
0x3	0x3	SINGLE	8-bit
0x4	0x4	SINGLE	8-bit
0x5	0x5	SINGLE	8-bit

CPU R/W Address[2:0]	PHADDR[2:0]	PHBURST	PHSIZE
0x6	0x6	SINGLE	8-bit
0x7	0x7	SINGLE	8-bit

6.2.4.2 Non-cacheable LDRH/STRH

CPU R/W Address[2:0]	PHADDR[2:0]	PHBURST	PHSIZE
0x0	0x0	SINGLE	16-bit
0x2	0x2	SINGLE	16-bit
0x4	0x4	SINGLE	16-bit
0x6	0x6	SINGLE	16-bit

6.2.4.3 Non-cacheable LDR/STR or LDM1/STM1

CPU R/W Address[2:0]	PHADDR[2:0]	PHBURST	PHSIZE
0x0	0x0	SINGLE	32-bit
0x4	0x4	SINGLE	32-bit

6.2.4.4 Non-cacheable LDM/STM2 ~ 16

CPU R/W Address[2:0]	Operation
0x0	2 ~ 16 single transfers No burst transfer
0x4	2 ~ 16 single transfers No burst transfer

Chapter 7

AXI Bus Interface Unit

This chapter contains the following sections:

7.1 AXI Control Signals

7.1.1 ID Signals

FA626TE does not drive any ID signal.

7.1.2 AxLEN[3:0]

7.1.2.1 I Master Port

I64 = 1

IARLEN[3:0]	Number of Data Transfer
b0000	1
b0001	2
b0010	3
b0011	4

I64 = 0

IARLEN[3:0]	Number of Data Transfer
b0000	1
b0001	2
b0010	3
b0011	4
b0100	5
b0101	6
b0110	7
b0111	8

7.1.2.2 RW Master Port

7.1.2.2.1 Address Read Channel

RW64 = 1

RWARLEN[3:0]	Number of Data Transfer
b0000	1
b0011	4

RW64 = 0

RWARLEN[3:0]	Number of Data Transfer
b0000	1
b0111	8

7.1.2.2.2 Address Write Channel

RW64 = 1

RWAWLEN[3:0]	Number of Data Transfer
b0000	1
b0001	2
b0010	3
b0011	4
b0100	5
b0101	6
b0110	7
b0111	8

RW64 = 0

RWAWLEN[3:0]	Number of Data Transfer
b0000	1
b0001	2
b0010	3
b0011	4

RWAWLEN[3:0]	Number of Data Transfer
b0100	5
b0101	6
b0110	7
b0111	8

7.1.2.3 P Master Port

7.1.2.3.1 Address Read Channel

P64 = 1

PARLEN[3:0]	Number of Data Transfer
b0000	1

P64 = 0

PARLEN[3:0]	Number of Data Transfer
b0000	1

7.1.2.3.2 Address Write Channel

P64 = 1

PAWLEN[3:0]	Number of Data Transfer
b0000	1

P64 = 0

PAWLEN[3:0]	Number of Data Transfer
b0000	1

7.1.2.4 DMA Master Port

7.1.2.4.1 Address Read Channel

DMAARLEN[3:0]	Number of Data Transfer
b0000	1
b0001	2
b0010	3
b0011	4
b0100	5
b0101	6
b0110	7
b0111	8
b1000	9
b1001	10
b1010	11
b1011	12
b1100	13
b1101	14
b1110	15
b1111	16

7.1.2.4.2 Address Write Channel

DMAAWLEN[3:0]	Number of Data Transfer
b0000	1
b0001	2
b0010	3
b0011	4
b0100	5
b0101	6
b0110	7
b0111	8

DMAAWLEN[3:0]	Number of Data Transfer
b1000	9
b1001	10
b1010	11
b1011	12
b1100	13
b1101	14
b1110	15
b1111	16

7.1.3 AxSIZE[2:0]

7.1.3.1 I Master Port

I64 = 1

IARSIZE[2:0]	Byte in Transfer
b011	8

I64 = 0

IARSIZE[2:0]	Byte in Transfer
b010	4

7.1.3.2 RW Master Port

7.1.3.2.1 Address Read Channel

RW64 = 1

RWARSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4
b011	8

RW64 = 0

RWARSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4

7.1.3.2.2 Address Write Channel

RW64 = 1

RWAWSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4
b011	8

RW64 = 0

RWAWSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4

7.1.3.3 P Master Port

7.1.3.3.1 Address Read Channel

P64 = 1

PARSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4

P64 = 0

PARSIZE[2:0]	Byte in transfer
b000	1
b001	2
b010	4

7.1.3.3.2 Address Write Channel

P64 = 1

PAWSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4

P64 = 0

PAWSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4

7.1.3.4 DMA Master Port

7.1.3.4.1 Address Read Channel

DMA64 = 1

DMAARSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4
b011	8

DMA64 = 0

DMAARSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4

7.1.3.4.2 Address Write Channel

DMA64 = 1

DMAAWSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4
b011	8

DMA64 = 0

DMAAWSIZE[2:0]	Byte in Transfer
b000	1
b001	2
b010	4

7.1.4 AxBURST[1:0]

7.1.4.1 I Master Port

IARBURST[1:0]	Burst Type
b01	Incr
b10	Wrap

7.1.4.2 RW Master Port

7.1.4.2.1 Address Read Channel

RWARBURST[1:0]	Burst Type
b01	Incr
b10	Wrap

7.1.4.2.2 Address Write Channel

RWAWBURST[1:0]	Burst Type
b01	Incr

7.1.4.3 P Master Port

7.1.4.3.1 Address Read Channel

PARBURST[1:0]	Burst Type
b01	Incr

7.1.4.3.2 Address Write Channel

PAWBURST[1:0]	Burst Type
b01	Incr

7.1.4.4 DMA Master Port

7.1.4.4.1 Address Read Channel

DMAARBURST[1:0]	Burst Type
b01	Incr

7.1.4.4.2 Address Write Channel

DMAAWBURST[1:0]	Burst Type
b01	Incr

7.1.5 AxLOCK[1:0]

7.1.5.1 I Master Port

IARLOCK[1:0]	Description
b00	Normal access

7.1.5.2 RW Master Port

7.1.5.2.1 Address Read Channel

RWARLOCK[1:0]	Description
b00	Normal access
b10	Locked access

7.1.5.2.2 Address Write Channel

RWAWLOCK[1:0]	Description
b00	Normal access

7.1.5.3 P Master Port

7.1.5.3.1 Address Read Channel

PARLOCK[1:0]	Description
b00	Normal access

7.1.5.3.2 Address Write Channel

PAWLOCK[1:0]	Description
b00	Normal access

7.1.5.4 DMA Master Port

7.1.5.4.1 Address Read Channel

DMAARLOCK[1:0]	Description
b00	Normal access

7.1.5.4.2 Address Write Channel

DMAAWLOCK[1:0]	Description
b00	Normal access

7.1.6 AxCACHE[3:0]/AxSIDEBAND[3:0]

AxCACHE supports the system-level caches.

AxSIDEBAND supports the internal caches.

7.1.6.1 I Master Port

IARCACHE[3:0]/IARSIDEBAND[3:0]	AXI Transaction Attribute	Memory Attribute C or B
b0000	NCNB	NCNB
b0001	Bufferable only	NCB
b0110	Cacheable Write-Through allocated on read only	CNB
b0111	Cacheable Write-Back allocated on read only	CB

7.1.6.2 RW Master Port

7.1.6.2.1 Address Read Channel

RWARSIDEBAND[3:0]/RWARCACHE[3:0]	AXI Transaction Attribute	Memory Attribute C or B
b0000	NCNB	NCNB
b0001	Bufferable only	NCB
b0110	Cacheable Write-Through allocated on read only	CNB
b1111 for RWARSIDEBAND[3:0] b0111 for RWARCACHE[3:0]	Cacheable Write-Back allocated on read and write for RWARSIDEBAND[3:0] Cacheable Write-Back allocated on read only for RWARCACHE[3:0]	CB

7.1.6.2.2 Address Write Channel

RWAWSIDEBAND[3:0]/RWAWCACHE[3:0]	AXI Transaction Attribute	Memory Attribute C or B	CR1-0 RAO Bit	CR1-0 STMWA Bit	STR/STM Cache Miss
b0000	NCNB	NCNB	-	-	-
b0001	Bufferable only	NCB	-	-	-
b0110	Cacheable Write-Through allocated on read only	CNB	-	-	-
b0110	Cacheable Write-Through allocated on read only	CB	1	-	-

RWAWSIDEBAND[3:0]/RWAWCACHE[3:0]	AXI Transaction Attribute	Memory Attribute C or B	CR1-0 RAO Bit	CR1-0 STMWA Bit	STR/STM Cache Miss
b1111 for RWAWSIDEBAND[3:0] b0111 for RWAWCACHE[3:0]	Cacheable Write-Back allocated on read and write for RWAWSIDEBAND[3:0] Cacheable Write-Back allocated on read only for RWAWCACHE[3:0]	CB	0	1	-
b0110	Cacheable Write-Through allocated on read only	CB	0	0	STM cache miss
b1111 for RWAWSIDEBAND[3:0] b0111 for RWAWCACHE[3:0]	Cacheable Write-Back allocated on read and write for RWAWSIDEBAND[3:0] Cacheable Write-Back allocated on read only for RWAWCACHE[3:0]	CB	0	0	STR cache miss

7.1.6.3 P Master Port

7.1.6.3.1 Address Read Channel

PARCACHE[3:0]/PARSIDEBAND[3:0]	AXI Transaction Attribute	Memory Attribute C or B
b0000	NCNB	NCNB
b0001	Bufferable only	NCB

7.1.6.3.2 Address Write Channel

PAWCACHE[3:0]/PAWSIDEBAND[3:0]	AXI Transaction Attribute	Memory Attribute C or B
b0000	NCNB	NCNB
b0001	Bufferable only	NCB

7.1.6.4 DMA Master Port

7.1.6.4.1 Address Read Channel

DMAARSIDEBAND[3:0]/DMAARCACHE[3:0]	AXI Transaction Attribute	Memory Attribute C or B
b0000	NCNB	NCNB
b0001	Bufferable only	NCB
b0110	Cacheable Write-Through allocated on read only	CNB
b1111 for DMAARSIDEBAND[3:0] b0111 for DMAARCACHE[3:0]	Cacheable Write-Back allocated on read and write for DMAARSIDEBAND[3:0] Cacheable Write-Back allocated on read only for DMAARCACHE[3:0]	CB

7.1.6.4.2 Address Write Channel

DMAAWSIDEBAND[3:0]/DMAAWCACHE[3:0]	AXI Transaction Attribute	Memory Attribute C or B
b0000	NCNB	NCNB
b0001	Bufferable only	NCB
b0110	Cacheable Write-Through allocated on read only	CNB
b1111 for DMAAWSIDEBAND[3:0] b0111 for DMAAWCACHE[3:0]	Cacheable Write-Back allocated on read and write for DMAAWSIDEBAND[3:0] Cacheable Write-Back allocated on read only for DMAAWCACHE[3:0]	CB

7.1.7 AxPROT[2:0]

7.1.7.1 I Master Port

IARPROT[2:0]	Description
IARPROT[2]	1 = Instruction access
IARPROT[1]	1 = Non-secure
IARPROT[0]	0 = User 1 = Privileged

7.1.7.2 RW Master Port

7.1.7.2.1 Address Read Channel

RWARPROT[2:0]	Description
RWARPROT[2]	0 = Data access
RWARPROT[1]	1 = Non-secure
RWARPROT[0]	0 = User 1 = Privileged

7.1.7.2.2 Address Write Channel

RWAWPROT[2:0]	Description
RWAWPROT[2]	0 = Data access
RWAWPROT[1]	1 = Non-secure
RWAWPROT[0]	0 = User 1 = Privileged

7.1.7.3 P Master Port

7.1.7.3.1 Address Read Channel

PARPROT[2:0]	Description
PARPROT[2]	0 = Data access
PARPROT[1]	1 = Non-secure
PARPROT[0]	0 = User 1 = Privileged

7.1.7.3.2 Address Write Channel

PAWPROT[2:0]	Description
PAWPROT[2]	0 = Data access
PAWPROT[1]	1 = Non-secure
PAWPROT[0]	0 = User 1 = Privileged

7.1.7.4 DMA Master Port

7.1.7.4.1 Address Read Channel

DMAARPROT[2:0]	Description
DMAARPROT[2]	0 = Data access
DMAARPROT[1]	1 = Non-secure
DMAARPROT[0]	0 = User 1 = Privileged

7.1.7.4.2 Address Write Channel

DMAAWPROT[2:0]	Description
DMAAWPROT[2]	0 = Data access
DMAAWPROT[1]	1 = Non-secure
DMAAWPROT[0]	0 = User 1 = Privileged

7.2 AXI Addressing Signals

7.2.1 I Master Port

7.2.1.1 Cacheable Fetch

I64 = 1

CPU R/W Address[4:0]	IARADDR[4:0]	IARBURST	IARSIZE	IARLEN
0x00	0x00	Incr	64-bit	Four data transfers
0x04	0x00	Incr	64-bit	Four data transfers
0x08	0x08	Wrap	64-bit	Four data transfers
0x0C	0x08	Wrap	64-bit	Four data transfers
0x10	0x10	Wrap	64-bit	Four data transfers
0x14	0x10	Wrap	64-bit	Four data transfers
0x18	0x18	Wrap	64-bit	Four data transfers
0x1C	0x18	Wrap	64-bit	Four data transfers

I64 = 0

CPU R/W Address[4:0]	IARADDR[4:0]	IARBURST	IARSIZE	IARLEN
0x00	0x00	Incr	32-bit	Eight data transfers
0x04	0x00	Incr	32-bit	Eight data transfers
0x08	0x08	Wrap	32-bit	Eight data transfers
0x0C	0x08	Wrap	32-bit	Eight data transfers
0x10	0x10	Wrap	32-bit	Eight data transfers
0x14	0x10	Wrap	32-bit	Eight data transfers
0x18	0x18	Wrap	32-bit	Eight data transfers
0x1C	0x18	Wrap	32-bit	Eight data transfers

7.2.1.2 Non-cacheable Fetch

I64 = 1

CPU R/W Address[4:0]	IARADDR[4:0]	IARBURST	IARSIZE	IARLEN
0x00	0x00	Incr	64-bit	Four data transfers
0x04	0x00	Incr	64-bit	Four data transfers
0x08	0x08	Incr	64-bit	Three data transfers
0x0C	0x08	Incr	64-bit	Three data transfers
0x10	0x10	Incr	64-bit	Two data transfers
0x14	0x10	Incr	64-bit	Two data transfers
0x18	0x18	Incr	64-bit	One data transfer
0x1C	0x18	Incr	64-bit	One data transfer

I64 = 0

CPU R/W Address[4:0]	IARADDR[4:0]	IARBURST	IARSIZE	IARLEN
0x00	0x00	Incr	32-bit	Eight data transfers
0x04	0x00	Incr	32-bit	Eight data transfers
0x08	0x08	Incr	32-bit	Six data transfers
0x0C	0x08	Incr	32-bit	Six data transfers
0x10	0x10	Incr	32-bit	Four data transfers
0x14	0x10	Incr	32-bit	Four data transfers
0x18	0x18	Incr	32-bit	Two data transfers
0x1C	0x18	Incr	32-bit	Two data transfers

7.2.2 RW Master Port

7.2.2.1 Address Read Channel

7.2.2.1.1 Line Fill

RW64 = 1

CPU R/W Address[4:0]	RWARADDR[4:0]	RWARBURST	RWARSIZE	RWARLEN
0x00 ~ 0x07	0x00	Incr	64-bit	Four data transfers
0x08 ~ 0x0F	0x08	Wrap	64-bit	Four data transfers
0x10 ~ 0x17	0x10	Wrap	64-bit	Four data transfers
0x18 ~ 0x1F	0x18	Wrap	64-bit	Four data transfers

RW64 = 0

CPU R/W Address[4:0]	RWARADDR[4:0]	RWARBURST	RWARSIZE	RWARLEN
0x00 ~ 0x03	0x00	Incr	32-bit	Eight data transfers
0x04 ~ 0x07	0x04	Wrap	32-bit	Eight data transfers
0x08 ~ 0x0B	0x08	Wrap	32-bit	Eight data transfers
0x0C ~ 0x0F	0x0C	Wrap	32-bit	Eight data transfers
0x10 ~ 0x13	0x10	Wrap	32-bit	Eight data transfers
0x14 ~ 0x17	0x14	Wrap	32-bit	Eight data transfers
0x18 ~ 0x1B	0x18	Wrap	32-bit	Eight data transfers
0x1C ~ 0x1F	0x1C	Wrap	32-bit	Eight data transfers

7.2.2.1.2 Non-cacheable LDRB

CPU R/W Address[2:0]	RWARADDR[2:0]	RWARBURST	RWARSIZE	RWARLEN
0x0	0x0	Incr	8-bit	One data transfer
0x1	0x1	Incr	8-bit	One data transfer
0x2	0x2	Incr	8-bit	One data transfer
0x3	0x3	Incr	8-bit	One data transfer
0x4	0x4	Incr	8-bit	One data transfer
0x5	0x5	Incr	8-bit	One data transfer
0x6	0x6	Incr	8-bit	One data transfer
0x7	0x7	Incr	8-bit	One data transfer

7.2.2.1.3 Non-cacheable LDRH

CPU R/W Address[2:0]	RWARADDR[2:0]	RWARBURST	RWARSIZE	RWARLEN
0x0	0x0	Incr	16-bit	One data transfer
0x2	0x2	Incr	16-bit	One data transfer
0x4	0x4	Incr	16-bit	One data transfer
0x6	0x6	Incr	16-bit	One data transfer

7.2.2.1.4 Non-cacheable LDR or LDM1

CPU R/W Address[2:0]	RWARADDR[2:0]	RWARBURST	RWARSIZE	RWARLEN
0x0	0x0	Incr	32-bit	One data transfer
0x4	0x4	Incr	32-bit	One data transfer

7.2.2.1.5 Non-cacheable LDM2 ~ 16/LDRD

CPU R/W Address[2:0]	Operation
0x0	2 ~ 16 single transfers No burst transfer
0x4	2 ~ 16 single transfers No burst transfer

7.2.2.2 Address Write Channel

7.2.2.2.1 Full-line Write-Back

RW64 = 1

CPU R/W Address[4:0]	RWAWADDR[4:0]	RWAWBURST	RWAWSIZE	RWAWLEN
Evicted cache line valid and both halves dirty	0x00	Incr	64-bit	Four data transfers

RW64 = 0

CPU R/W Address[4:0]	RWAWADDR[4:0]	RWAWBURST	RWARSIZE	RWARLEN
Evicted cache line valid and both halves dirty	0x00	Incr	32-bit	Eight data transfers

7.2.2.2.2 Half-line Write-Back

RW64 = 1

CPU R/W Address[4:0]	RWAWADDR[4:0]	RWAWBURST	RWAWSIZE	RWAWLEN
Evicted cache line valid and lower half dirty	0x00	Incr	64-bit	Two data transfers
Evicted cache line valid and higher half dirty	0x10	Incr	64-bit	Two data transfers

RW64 = 0

CPU R/W Address[4:0]	RWAWADDR[4:0]	RWAWBURST	RWAWSIZE	RWAWLEN
Evicted cache line valid and lower half dirty	0x00	Incr	32-bit	Four data transfers
Evicted cache line valid and higher half dirty	0x10	Incr	32-bit	Four data transfers

7.2.2.2.3 Cacheable Write-Through or Non-cacheable STRB

RW64 = 1

CPU R/W Address[2:0]	RWAWADDR[2:0]	RWAWBURST	RWAWSIZE	RWAWLEN	RWWSTRB
0x0	0x0	Incr	8-bit	One data transfer	b0000 0001
0x1	0x1	Incr	8-bit	One data transfer	b0000 0010
0x2	0x2	Incr	8-bit	One data transfer	b0000 0100
0x3	0x3	Incr	8-bit	One data transfer	b0000 1000
0x4	0x4	Incr	8-bit	One data transfer	b0001 0000
0x5	0x5	Incr	8-bit	One data transfer	b0010 0000
0x6	0x6	Incr	8-bit	One data transfer	b0100 0000
0x7	0x7	Incr	8-bit	One data transfer	b1000 0000

RW64=0

CPU R/W Address[2:0]	RWAWADDR[2:0]	RWAWBURST	RWAWSIZE	RWAWLEN	RWWSTRB
0x0	0x0	Incr	8-bit	One data transfer	b0001
0x1	0x1	Incr	8-bit	One data transfer	b0010
0x2	0x2	Incr	8-bit	One data transfer	b0100
0x3	0x3	Incr	8-bit	One data transfer	b1000
0x4	0x4	Incr	8-bit	One data transfer	b0001
0x5	0x5	Incr	8-bit	One data transfer	b0010
0x6	0x6	Incr	8-bit	One data transfer	b0100
0x7	0x7	Incr	8-bit	One data transfer	b1000

7.2.2.2.4 Cacheable Write-Through or Non-cacheable STRH

RW64 = 1

CPU R/W Address[2:0]	RWAWADDR[2:0]	RWAWBURST	RWAWSIZE	RWAWLEN	RWWSTRB
0x0	0x0	Incr	16-bit	One data transfer	b0000 0011
0x2	0x2	Incr	16-bit	One data transfer	b0000 1100
0x4	0x4	Incr	16-bit	One data transfer	b0011 0000
0x6	0x6	Incr	16-bit	One data transfer	b1100 0000

RW64 = 0

CPU R/W Address[2:0]	RWAWADDR[2:0]	RWAWBURST	RWAWSIZE	RWAWLEN	RWWSTRB
0x0	0x0	Incr	16-bit	One data transfer	b0011
0x2	0x2	Incr	16-bit	One data transfer	b1100
0x4	0x4	Incr	16-bit	One data transfer	b0011
0x6	0x6	Incr	16-bit	One data transfer	b1100

7.2.2.2.5 Cacheable Write-Through or Non-cacheable STR or STM1

RW64 = 1

CPU R/W Address[2:0]	RWAWADDR[2:0]	RWAWBURST	RWAWSIZE	RWAWLEN	RWWSTRB
0x0	0x0	Incr	32-bit	One data transfer	b0000 1111
0x4	0x4	Incr	32-bit	One data transfer	b1111 0000

RW64 = 0

CPU R/W Address[2:0]	RWAWADDR[2:0]	RWAWBURST	RWAWSIZE	RWAWLEN	RWWSTRB
0x0	0x0	Incr	32-bit	One data transfer	b1111
0x4	0x4	Incr	32-bit	One data transfer	b1111

7.2.2.2.6 Cacheable Write-Through or Non-cacheable STM2 ~ 16/STRD

CPU R/W Address[4:0]	Operation
0x00	If STM is in the NCNB region, it will always issue multiple single writes with a length of 1; otherwise, STM will be split into several burst transfers depending on available data in the write buffer. Each time when the bus interface issues a write command, it will merge the continuous data in the write buffer into a burst write.
0x04	

When RW64 = ‘1’, some two 32-bit may be merged into one 64-bit data.

7.2.2.3 Multiple Cacheable Write-Through or Non-cacheable STRB/STRH/STR/STRD with continuous address.

It is the same cases as Cacheable Write-Through or Non-cacheable STM2~16/STRD.

7.2.3 P Master Port

7.2.3.1 Non-cacheable LDRB/STRB

P64 = 1

CPU R/W Address[4:0]	PARADDR[4:0]/ PAWADDR[4:0]	PARBURST/ PAWBURST	PARSIZE/ PAWSIZE	PARLEN/ PAWLEN	PWSTRB
0x00	0x00	Incr	8-bit	One data transfer	b0000 0001
0x01	0x01	Incr	8-bit	One data transfer	b0000 0010
0x02	0x02	Incr	8-bit	One data transfer	b0000 0100
0x03	0x03	Incr	8-bit	One data transfer	b0000 1000
0x04	0x04	Incr	8-bit	One data transfer	b0001 0000
0x05	0x05	Incr	8-bit	One data transfer	b0010 0000
0x06	0x06	Incr	8-bit	One data transfer	b0100 0000
0x07	0x07	Incr	8-bit	One data transfer	b1000 0000

P64 = 0

CPU R/W Address[4:0]	PARADDR[4:0]/ PAWADDR[4:0]	PARBURST/ PAWBURST	PARSIZE/ PAWSIZE	PARLEN/ PAWLEN	PWSTRB
0x00	0x00	Incr	8-bit	One data transfer	b0001
0x01	0x01	Incr	8-bit	One data transfer	b0010
0x02	0x02	Incr	8-bit	One data transfer	b0100
0x03	0x03	Incr	8-bit	One data transfer	b1000
0x04	0x04	Incr	8-bit	One data transfer	b0001
0x05	0x05	Incr	8-bit	One data transfer	b0010
0x06	0x06	Incr	8-bit	One data transfer	b0100
0x07	0x07	Incr	8-bit	One data transfer	b1000

7.2.3.2 Non-cacheable LDRH/STRH

P64 = 1

CPU R/W Address[4:0]	PARADDR[4:0]/ PAWADDR[4:0]	PARBURST/ PAWBURST	PARSIZE/ PAWSIZE	PARLEN/ PAWLEN	PWSTRB
0x00	0x00	Incr	16-bit	One data transfer	b0000 0011
0x02	0x02	Incr	16-bit	One data transfer	b0000 1100
0x04	0x04	Incr	16-bit	One data transfer	b0011 0000
0x06	0x06	Incr	16-bit	One data transfer	b1100 0000

P64 = 0

CPU R/W Address[4:0]	PARADDR[4:0]/ PAWADDR[4:0]	PARBURST/ PAWBURST	PARSIZE/ PAWSIZE	PARLEN/ PAWLEN	PWSTRB
0x00	0x00	Incr	16-bit	One data transfer	b0011
0x02	0x02	Incr	16-bit	One data transfer	b1100
0x04	0x04	Incr	16-bit	One data transfer	b0011
0x06	0x06	Incr	16-bit	One data transfer	b1100

7.2.3.3 Non-cacheable LDR/STR or LDM1/STM1

P64 = 1

CPU R/W Address[4:0]	PARADDR[4:0]/ PAWADDR[4:0]	PARBURST/ PAWBURST	PARSIZE/ PAWSIZE	PARLEN/ PAWLEN	PWSTRB
0x00	0x00	Incr	32-bit	One data transfer	b0000 1111
0x04	0x04	Incr	32-bit	One data transfer	b1111 0000

P64 = 0

CPU R/W Address[4:0]	PARADDR[4:0]/ PAWADDR[4:0]	PARBURST/ PAWBURST	PARSIZE/ PAWSIZE	PARLEN/ PAWLEN	PWSTRB
0x00	0x00	Incr	32-bit	One data transfer	b1111
0x04	0x04	Incr	32-bit	One data transfer	b1111

7.2.3.4 Non-cacheable LDM/STM2 ~ 16

CPU R/W Address[4:0]	Operations
0x00	2 ~ 16 single transfers No burst transfer
0x04	2 ~ 16 single transfers No burst transfer

Chapter 8

Scratchpads

This chapter contains the following sections:

In order to obtain more predictable program behaviors and improve the overall system performance, FA626TE incorporates an instruction scratchpad (IScratchpad) and a data scratchpad (DScratchpad) in its design. The current design is 8K byte each. IScratchpad and DScratchpad can be mapped to any location in the memory address space. The base addresses of IScratchpad and DScratchpad should be programmed at the CR9 register by using the physical address. Both scratchpads have programmable base addresses and sizes.

Programmers can set up the base and size configuration by the following instruction.
MCR p15, 0, Rd, c9, c1, op2

where “op2” must be ‘0’ for DScratchpad, and ‘1’ for IScratchpad. The Rd register contains the base/size and enable/disable configurations. Please refer to Chapter 2 for more information.

On reset, both IScratchpad and DScratchpad are disabled.

8.1 Instruction Scratchpad (IScratchpad)

After setting the base and size configuration and enabling IScratchpad, IScratchpad will wait for the starting of an auto-fill stage. When an instruction fetch falls in the address range of IScratchpad at the first time, the IScratchpad will enter the auto-fill stage and start fetching instructions from the external memory.

During the fetching stage of IScratchpad, FA626TE will be temporarily stalled until the instructions are fetched and stored in the full-size IScratchpad. After being filled, all instructions whose addresses fall into the region of the IScratchpad will be fetched from IScratchpad.

If programmers want to fill other codes into IScratchpad, IScratchpad must be flushed first. The following instruction can be used to flush all IScratchpads.
MCR p15, 0, Rd, C7, C5, 5

The content of Rd register should be zero.

8.2 Data Scratchpad (DScratchpad)

After setting the base and size configuration and enabling DScratchpad, DScratchpad will be valid and ready to serve the memory access immediately. All the data accesses that fall in the memory range of DScratchpad will be targeted to DScratchpad.

Programmers can store temporary data in DScratchpad for future operations. Because there is no auto-fill mechanism for DScratchpad, programmers are required to manually fill the data to DScratchpad. All the memory access instructions can be used to access DScratchpad.

There is no invalidated instruction for DScratchpad.

DScratchpad should be used as a temporary space for data manipulations. If the base/size definition of DScratchpad is changed and the data in DScratchpad needs to be preserved for future use, programmers must transfer data in DScratchpad and save data to the external memory using normal store instruction. The initial content of DScratchpad is not defined. A load access to a location in DScratchpad prior to a store access to the same location will result in no initialized value that can be read into DScratchpad.

DScratchpad is a single-port SRAM. FA626TE provides a dedicated DMA slave port for DScratchpad to perform the high-speed data transfer. CPU has higher priority over the external DMA device.

Chapter 9

Reset and Clocking Control (Power-down Control)

This chapter contains the following sections:

9.1 Clocking Modes of FA626TE Core

FA626TE supports the synchronous clocking modes. Better skew control is needed on the logic that generates the clock synchronization input.

9.1.1 Clock Synchronization

In the synchronous mode, the clock synchronization signal (BCKEN) is simply a usual clock enable signal. This signal indicates the effective CPU clock edge related to the AHB bus clock.

Figure 9-1 depicts the relationship between BCLKEN and HCLK.

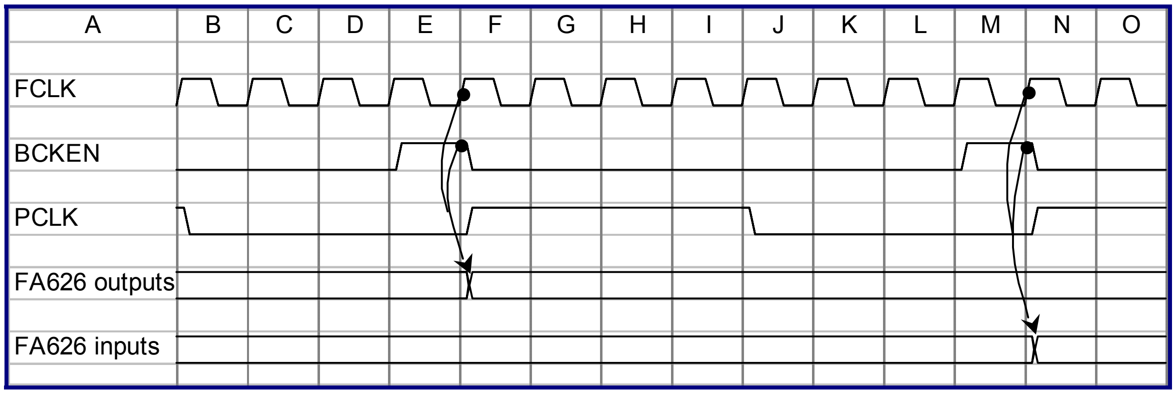

Figure 9-1. Timing of Clock Enable Signal

9.2 Reset of FA626TE Cores

The FA626TE cores need at least two CPU cycles for the reset assertion. When resetting the de-assertion state, the first bus request will be issued on the bus after 390 CPU cycles. Please refer to the following timing diagram for details.

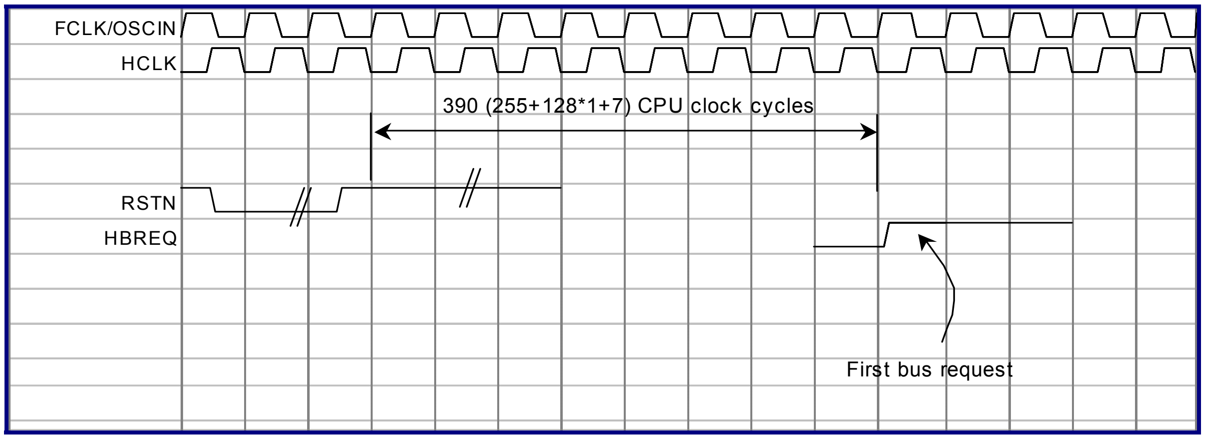

Figure 9-2. FA626TE Reset Timing

9.3 Power-saving Modes

FA626TE supports the power-saving mode. To enter the power-saving mode, users need to execute the idle instruction (MCR p15, 0, Rd, C7, C0, 4). FA626TE will enter the idle mode when no pipeline is stalled, no cache is missing, and write buffer is empty. When CPU enters the power-saving mode, the processor will stop the instruction execution and stay in an idle state. The system can then stop the CPU input clock to save power. The idle instruction cannot be executed in the user mode. To exit from the idle mode, the CPU clock must start first, and then both external interrupts (FIQ and IRQ) and command from the ICE port can wake up the processor from the power-saving mode.

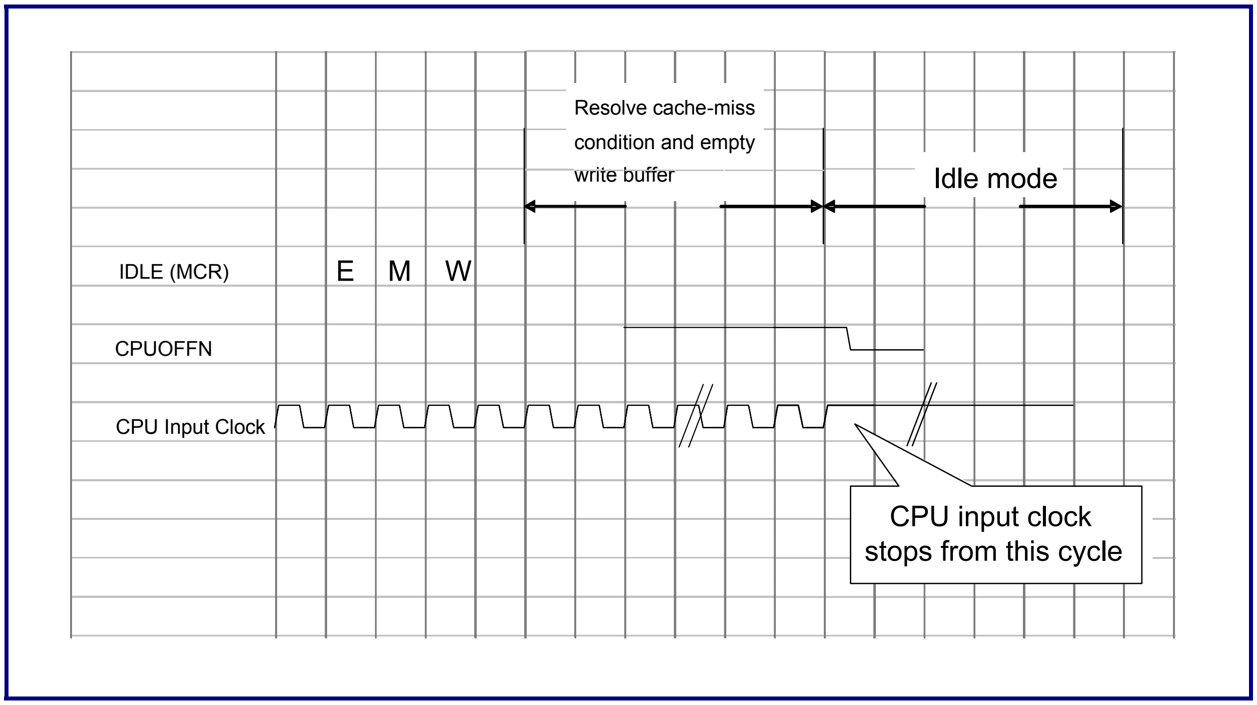
Resolve cache-miss condition and empty write buffer
Idle mode
IDLE (MCR)
E
M
W
CPUOFFN
CPU Input Clock
CPU input clock stops from this cycle

Figure 9-3. CPU Entering Idle Mode and System Stopping CPU Clock

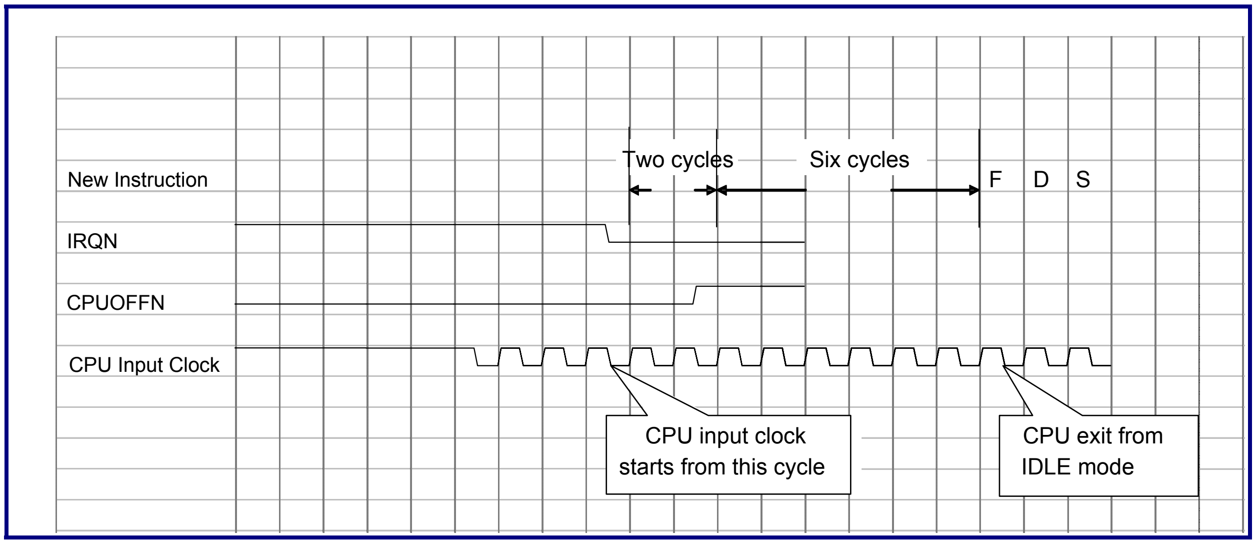

Figure 9-4. Leaving Idle Mode

If the processor is woken up by an external interrupt (IRQ or FIQ), it will resume the program execution following the idle instruction and after a return-from-interrupt instruction (SUBS PC, R14, #4) in the executing interrupt service routine. If the processor is woken up through the ICE port, it will enter the power-saving mode after exiting from the ICE state.

Chapter 10

ARM ICE Debug Facility

This chapter contains the following sections:

10.1 Debug Systems

The ARM ICE debug facility is supported by FA626TE, which provides a debug protocol connecting REALVIEW™ ICE. This debug system is comprised of three major components: The debug host, protocol converter, and debug target. The system diagram is shown in Figure 10-1.

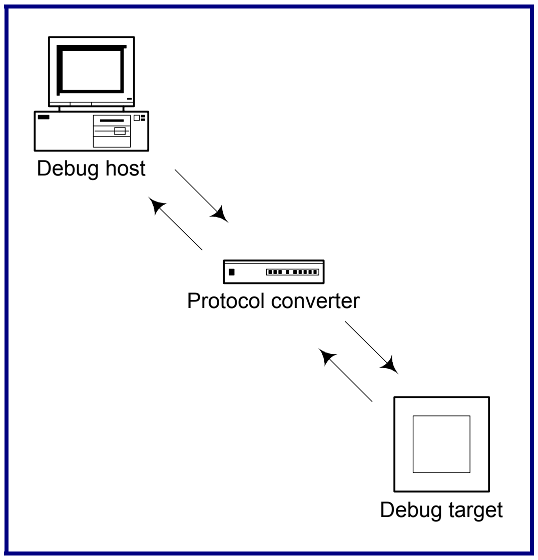

Figure 10-1. Debug System Diagram

10.1.1 Debug Host

The debug host is a computer equipped with a software debugger, such as the AXD™ or RealView™ debugger, and is used to issue the high-level commands, such as the breakpoint setting or the memory content examination.

10.1.2 Protocol Converter

The protocol converter is used to translate the commands issued by the debug host to the JTAG control signals used by the debug target.

10.1.3 Debug Target

The debug target is the Faraday FA626TE processor. The debug functions include:

- Stalling processor from program execution by breakpoint, watchpoint, and asynchronous debug request
- Examining or modifying processor state
- Examining or modifying system state that is reachable through system bus
- Resuming program execution

10.2 Entry into Debug State

There are three cases to force the processor into the debug state:

- Breakpoint match
- Watchpoint match
- Immediate debug request

There are two sets of hardware breakpoint or watchpoint registers in FA626TE. Each set can be used as either the breakpoint or watchpoint according to the software setting. There is another control register in FA626TE for generating immediate debug requests. The following section shows the timing of entering the debug state for the three cases listed above.

10.2.1 Breakpoint Match

When a breakpoint is enabled, the debug unit will start monitoring the program counter and instruction content. Once a match is found, the core will start entering the debug state. The following diagram explains the timing of entering the debug state by a breakpoint match.

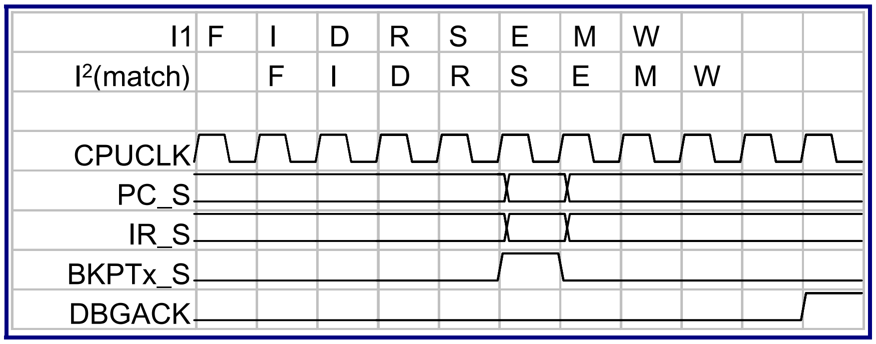

Figure 10-2. Breakpoint Match Timing

The breakpoint is detected at the Shift stage of the processor pipeline. The debugger entering sequence will start when the matched instruction enters the Write-Back stage of the pipeline. The instructions executed before the matched instruction will be completed. The matched instruction and instructions subsequent to the matched instruction will be canceled. Extra clock cycles may be needed before entering the debug state to wait for the idle states of ICache and DCache.

If an exception occurs simultaneously with a breakpoint, the exception has higher priority than the breakpoint. The processor will jump to the exception handler and the breakpoint will be discarded. When the program execution is returned from the exception handler, the breakpoint will then take effect when the breakpoint is matched again.

10.2.2 Watchpoint Match

When a watchpoint is enabled, the debug unit will start monitoring the data address and store value (The read value will not be monitored). Once a match is found, the core will start entering the debug state. The following diagram explains the timing of entering the debug state by watchpoint match:

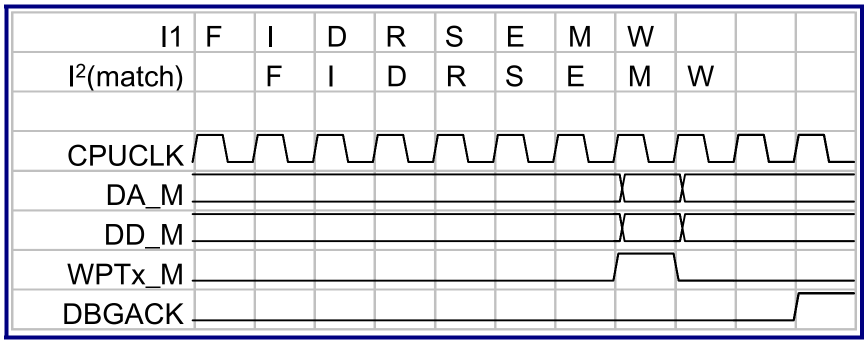

Figure 10-3. Watchpoint Match Timing

The watchpoint is compared at the Memory-Access stage of processor pipeline. The debug entering sequence will start when the matched instruction enters the Write-Back stage of the pipeline. The instructions before the matched instruction will be completed. The matched instruction and instructions after the matched instruction will be canceled. Extra clock cycles may be needed before entering the debug state to wait for the idle states of ICache and DCache.

The read value is returned at the end of M stage when DCache hits; however, if DCache misses, the data return timing will be unpredictable (Hit-under-miss feature). Consequently, the read value will not be monitored.

If an exception occurs simultaneously with a watchpoint, the exception has higher priority over the watchpoint. The processor will jump to the exception handler and the watchpoint will be discarded. If the program execution is returned from the exception handler, the watchpoint will then take effect when the watchpoint is matched again.

10.2.3 Immediate Debug Request

The immediate debug request can be issued by writing ‘1’ to the DBGRQ bit in the Debug Control Register. The processor enters the debug state when the instruction at the current execution stage is completed.

If an exception occurs simultaneously with an immediate debug request, the exception has higher priority over the immediate debug request. The processor will stop at the program counter of the first instruction in the exception handler. The first instruction of the exception handler will not be executed.

10.2.4 Behavior of FA626TE in Debug State

When the FA626TE is in the debug state, both ICache and DCache will be disconnected from the processor core. Any memory request from the processor core will be ignored by ICache and DCache. In the ICE mode, the processor core will ignore the permission check but the other aborts will be recorded in FAR and FSR without data abort trap. The processor core will also ignore the FIQ and IRQ interrupts in debug state.

10.3 Scan Chains and JTAG Interface

Inside the FA626TE processor, there are three scan chains used for processor debugging or ICE module programming. These scan chains are scan chains 1, 2, and 15. A JTAG-style Test Access Port (TAP) controller is used to control these scan chains as shown in Figure 10-4. The TAP controller can support up to 32 scan chains.

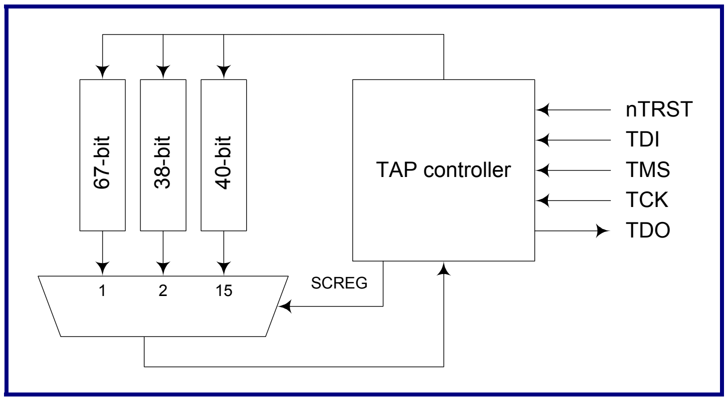

Figure 10-4. TAP Controller

10.3.1 Public TAP Instruction

The FA626TE processor supports five TAP instructions. These instructions are listed in Table 10-1.

Table 10-1. TAP Instructions Supported by FA626TE Processor

Instruction	Binary Code	Applied Scan Chain
SCAN_N	0010	SCREG register
INTEST	1100	Scan chains 1,2,15
IDCODE	1110	IDREG register
BYPASS	1111	BPREG register
RESTART	0100	BPREG register

The following are the TAP instructions supported by the FA626TE processor:

- **SCAN_N (0010)**

  This instruction is used to change the scan chain select register (SCREG) and to connect the SCREG between TDI and TDO. The width of the SCREG register is 5-bit.

- **INTEST (1100)**

  This instruction is used to connect the selected scan chain between TDI and TDO.

- **IDCODE (1110)**

  This instruction is used to obtain value from the JTAG device identification register (ID register) of the debug target. The debugger will use the ID code to identify the debug target.

- **BYPASS (1111)**

  This instruction is used to connect a 1-bit bypass register between TDI and TDO. The data shifted into the bypass register will be shifted out from TDO after a delay of one TCK cycle.

- **RESTART (0100)**

  This instruction is used to restart the FA626TE processor execution when exiting from the debug state.

10.3.2 Supported Scan Chains

The FA626TE processor supports three scan chains. The functions of these scan chains are listed in Table 10-2.

Table 10-2. Scan Chains Supported by FA626TE Processor

Scan Chain Number	Length	Function
1	67	Debug
2	38	ICE module programming
15	40	Coprocessor 15

The following are the scan chains supported by the FA626TE processor:

- **Scan Chain 1**

  Scan chain 1 is primarily for debugging. This scan chain is 67-bit long, with 32 bits for instruction data, 32 bits for data value, two control bits and one reserved bit. Table 10-3 lists the function of each bit of this scan chain. The detailed behavior of this scan chain is described below. Scan chain 1 only supports the INTEST command.

  - The instruction data field provides processor with the instructions to be executed in debug state.
  - The data value field provides processor with the read data in debug state.
  - After the processor enters debug state and at the first time of reading this scan chain, the SYSSPEED bit indicates which event, breakpoint or watchpoint, causes the debug entrance. Otherwise, users can write a 1 to the SYSSPEED bit to force the FA626TE processor to go back to system speed before executing the instruction.
  - The WPTANDBKPT bit is used to indicate if the breakpoint event and watchpoint event happen at the same time.
  - In Capture-DR state, if there is a store or store-multiple instruction in the M stage of pipeline, the content on DD bus will be captured into the data value field of this scan chain.

Table 10-3. Bit Functions of Scan Chain 1

Bit	Function
[0:31]	Instruction data ID[0:31]
32	WPT (At first read)/SYSSPEED
33	WPTANDBKPT
34	Reserved
[35:66]	Data value DD[31:0]

- **Scan Chain 2**

  Scan chain 2 is primarily for programming the ICE module. This scan chain is 38-bit long, with 32 bits for data value, 5 bits for addressing ICE register, and 1 bit for read/write control. The bit function of this scan chain is listed in Table 10-4. The detailed structure is shown in Figure 10-5. Please note that Scan chain 2 only supports the INTEST command.

Table 10-4. Bit Functions of Scan Chain 2

Bit	Function
0	nR/W 0/1 ←→ Read/Write
[1:5]	ICE register address
[6:37]	Data value

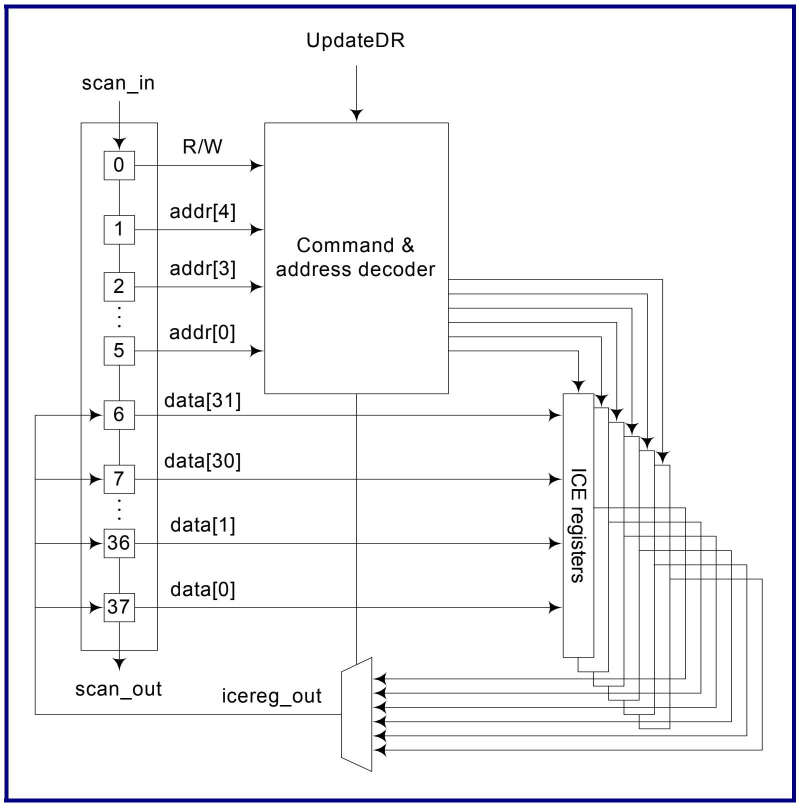

Figure 10-5. Scan Chain 2 Structure

- **Scan Chain 15**

  Scan chain 15 is for accessing the system control registers in CP15 and performing special system operations to ICache, DCache, MMU, write buffer, and so on. This scan chain is 40-bit long, with 32 bits for data value or instruction word, 6 bits for addressing CP15 register, 1 bit for read/write control and 1 bit for mode selection. Scan chain 15 offers two operation modes: Physical access mode and interpreted access mode. The bit function of this scan chain is listed in Table 10-5. Please note that Scan chain 15 only supports the INTEST commands.

Table 10-5. Bit Functions of Scan Chain 15

Bit	Interpreted Access Mode		Physical Access Mode	
	Function	R/W	Function	R/W
0	0	W	nR/W	W
[1:6]	000000	W	Register address	W
[7:38]	Instruction word	W	Register value	R/W
39	0	W	1	W

Physical Access Mode

This mode is used to directly access the system control registers in CP15. The data will be unchanged if the control register is read-only. Table 10-6 lists the accessible registers in CP15 of the FA626TE processor and their address mapping.

Table 10-6. Address Mapping of CP15 Register in Physical Access Mode

Register Address			Register Number	Name	Type
[1]	[2:5]	[6]			
0	0x0	0	C0	ID code register	Read-only
0	0x0	1	C0	Cache type register	Read-only
0	0x1	0	C1	Configuration register	R/W
0	0x2	0	C2	Translation table base register	R/W
0	0x3	0	C3	Domain access control register	R/W
0	0x5	0	C5	Data fault status register (DFSR)	R/W
0	0x5	1	C5	Instruction fault status register (IFSR)	R/W
0	0x6	0	C6	Fault address register	R/W
0	0x9	0	C9	DCache lockdown register	R/W
0	0xa	0	C10	TLB lockdown register (TLBL)	Write-only
0	0xd	0	C13	Process ID register	R/W
0	0xf	0	C15	Test state register	R/W

Interpreted Access Mode

The process of the interpreted access mode operation is listed below:

1. Set bit 0 of the CP15 test state register through the physical access mode to enable the interpreted access mode.
2. Shift the intended MRC/MCR instruction into scan chain 15 through the interpreted access mode.
3. Perform a system-speed LDR/STR instruction on the FA626TE processor.
4. In the LDR case, the data value from the coprocessor register is stored to the register of the FA626TE processor specified by the LDR instruction.
5. In the STR case, the stored data value is passed to the interpreted MCR instruction to be stored to the coprocessor register.
6. Clear bit 0 of the CP15 test state register through the physical access mode to disable the interpreted access mode.

The register Rd of the MRC and MCR instructions is a dummy in the interpreted access mode. The data accessed from CP15 through an MRC instruction is passed to the Rd register specified in the LDR instruction. The data value stored to CP15 through an MCR instruction is from the register specified in the Rd register of the STR instruction.

The mapping of the MCR/MRC instructions to the CP15 registers in the interpreted access mode is listed in Table 10-7.

Table 10-7. Mapping of MCR/MRC Instructions to CP15 Registers in Interpreted Access Mode

FA626TE Instruction	Function	Rd	Ra	CP15 Instruction
STR Rd, [Ra]	Write TTB	TTB	-	MCR p15, 0, r0, c2, c0, 0
LDR Rd, [Ra]	Read TTB	TTB	-	MRC p15, 0, r0, c2, c0, 0

10.4 Determining the Processor State and System State

When the FA626TE processor is in the debug state, programmers can use the processor instructions to examine the processor state and system state. The instructions are shifted into the processor pipeline through the scan chain.

10.4.1 Determining the Processor State

When the FA626TE processor is in the debug state, programmer can use the processor instructions to examine the processor state. Table 10-8 lists the typical instructions used to determine the processor state.

Table 10-8. Instructions for Determining Processor State

Instruction	Description
All data processing	The generated necessary data
STM, STR	Get register contents from current bank
LDM, LDR	Set register contents to current bank
MRS	Get CPSR register content
MSR	Set CPSR register content

Programmers can use the MRS/MSR instructions to change the mode for accessing other register banks. All the above instructions are executed at the debug speed. DCache and ICache are disconnected from the processor core while using these instructions to determine the processor state.

10.4.2 Determining the System State

With the dynamic timing requirement of memory system, the processor is required to be run at the system speed to examine the system state. The instructions used for determining the system state are mostly the same as those used for determining the processor state. The difference is that the load and store instructions used to access the system state need to be executed at the system speed via setting the SYSSPEED bit to ‘1’. DCache will be connected to the processor while a system speed load or store instruction is executed. Only the load and store instructions can be executed at the system speed. After a system speed access is completed and the processor goes back to the debug state, the SYSSPEED bit will be set to high.

The sequence of executing a system speed instruction is shown below:

1. Put a load or store instruction in the ID field of Scan Chain 1 by setting the SYSSPEED bit to ‘0’. Push the pipeline one step forward.
2. Put an NOP instruction in the ID field of Scan Chain 1 by setting the SYSSPEED bit to ‘1’. Push the pipeline one step forward.
3. Load a RESTART command into the TAP controller and force the TAP controller into the RUN-TEST/IDLE state. This will cause the FA626TE processor to execute instructions at the system speed.

The FA626TE processor will return to the debug state if DBGRQ (Bit 1 of the debug control register) remains high. SYSCOMP (Bit 3 of the debug status register) will be set to ‘1’ if the system speed instruction is completed. The returned system memory content can be passed to the debug host by executing an STM instruction at the debug speed.

10.5 Exit from Debug State

When the processor enters the debug state, the debugger should back up all the processor states before performing any debug operation. These processor states are restored before exiting from the debug state. Programmers can exit the debug state by using the branch instruction. This instruction will resume the program execution back to the instruction that is suspended before entering the debug state. The calculation of the resumed program counter value for the branch is described in the next section. The following sequence is used to exit from the debug state:

1. Restore the processor state. The PC (R15) should be updated.
2. Clear the DBGRQ and DBGACK bits in the debug control register
3. Clock the “B -n” instruction into the pipeline and set the SYSSPEED bit to low. The value of n depends on the event that triggers the debug entrance.
4. Clock the NOP instruction into the pipeline and set the SYSSPEED bit to high.
5. Load a RESTART command into the TAP controller and force the TAP controller into the Run-Test/Idle state.

When the TAP instruction is RESTART and the TAP controller is entering the Run-Test/Idle state, a one-clock-cycle RESTART pulse will be generated by the TAP controller. The ICache and DCache will be connected to the processor core after the RESTART pulse.

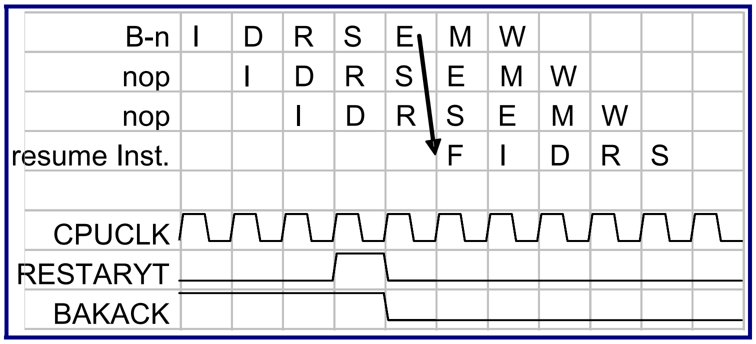

Figure 10-6. Exiting Debug Mode

10.6 Behavior of the Program Counter during Debug

The program counter calculation is the same for breakpoint, watchpoint, and immediate debug request. The returned address is calculated by the following formula:

-(2 + N + 6S)

Where "N" is the number of instructions executed at the debug speed and "S" is the number of instructions executed at the system speed. Several examples are illustrated below.

Entering the Debug State from the Breakpoint Instruction

If no instruction is executed in the debug state, use the following instructions to exit the debug state and resume the next instruction.

1. B -3 with SYSSPEED bit low
2. NOP with SYSSPEED bit high

The number "-3" is the result of -(2 + 1). The number "N" is 1 for the branch instruction (B -3) itself. The NOP instruction executed at the system speed is not counted.

Entering the Debug State from the Watchpoint Instruction

Same as the above example

Entering the Debug State from the Immediate Debug Request

Same as the above example

Backup the Program Counter at the Debug State

If users read the program counter by using the store instruction in the debug state, the returned program counter will be four words ahead of the breakpoint instruction. For example, if the breakpoint is set at address 0x238, the returned program counter will be 0x248. The debugging tool can save the returned program counter value for resuming the program execution.

If this is the case, after the saved program counter value is restored back to the program counter through the debug speed instruction, the following instructions can be used to exit the debug state and resume the next instruction:

1. B -6 with the SYSSPEED bit low
2. NOP with the SYSSPEED bit high

The number “–6” is the result of -(2 + 4). The number “N” is four, one for the branch instruction (B -6) itself and three for the operation of read program counter. The NOP instruction executed at system speed is not counted.

10.7 ICE Operation

Access the CPU Registers

1. Scan a load/store/ldm/stm instruction into the ID field of Scan Chain 1 with the SYSSPEED bit set to ‘0’, and push the ID into pipeline (One TCK at Run-Test/Idle state).
2. Scan an NOP instruction into the ID field of Scan Chain 1 with the SYSSPEED bit set to ‘0’, and push the ID into pipeline (Four TCKs at Run-Test/Idle state) for pushing the load/store/ldm/stm instruction into the M stage of the pipeline.
3. Read/Write the DD field for accessing the data of CPU registers, one TCK in Run-Test/Idle state for writing data to registers or for reading data from next registers.

If there are other registers to be accessed, repeat Step 3 until all register accesses are complete.

Change Mode

1. Scan an MSR instruction into the ID field of Scan Chain 1 with the SYSSPEED bit set to ‘0’, and push the ID into pipeline (One TCK in Run-Test/Idle state).
2. Scan an NOP instruction into the ID field of Scan Chain 1 with the SYSSPEED bit set to ‘0’, and push the ID into pipeline (Five TCKs in Run-Test/Idle state).

Read PC while Entering ICE Mode

The returned address is five words in advance of the stopped instruction (Using STM, STR instruction).

Exit the Debug State after PC is Restored

B - N - 1

Exit the Debug State without Executing Any Instruction

B -4

10.8 ICE Module

The FA626TE ICE module offers the functions of debug control, debug status, breakpoint, watchpoint, and debug communication channel. There are two monitoring units in the ICE module. Each unit can be programmed to monitor a breakpoint or a watchpoint. Each unit consists of an address register, a data register, and a control register. Each of the three registers has a corresponding mask register. The breakpoint can be set to monitor any program address, program address range, certain instruction content, or certain instruction fields. The watchpoint can be set to monitor the data address, data address range, data content, or data fields with the data widths or the read/write directions. The data paths to generate the breakpoint and watchpoint are shown in Figure 10-7. Please note that, the watchpoint does not support the data content comparison on memory read instructions.

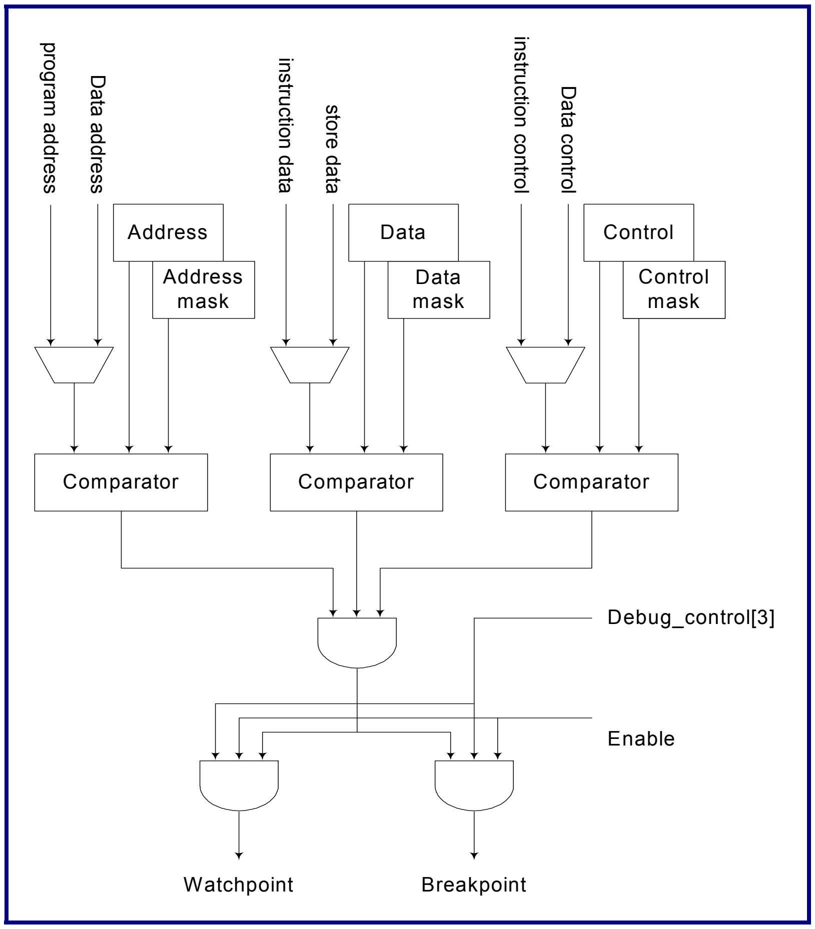

Figure 10-7. Breakpoint and Watchpoint Generations

Register Map

The mapping of the ICE module register is listed in Table 10-9.

Table 10-9. Mapping of ICE Module Register

Address	Width	Descriptiion	Type
00000	4	Debug control	R/W
00001	5	Debug status	Read-only
00010	8	Vector catch control	R/W
00100	6	Debug communication control	Read-only
00101	32	Debug communication data	R/W
01000	32	Address value of watchpoint 0	R/W
01001	32	Address mask of watchpoint 0	R/W
01010	32	Data value of watchpoint 0	R/W
01011	32	Data mask of watchpoint 0	R/W
01100	9	Control value of watchpoint 0	R/W
01101	8	Control mask of watchpoint 0	R/W
10000	32	Address value of watchpoint 1	R/W
10001	32	Address mask of watchpoint 1	R/W
10010	32	Data value of watchpoint 1	R/W
10011	32	Data mask of watchpoint 1	R/W
10100	9	Control value of watchpoint 1	R/W
10101	8	Control mask of watchpoint 1	R/W

10.8.1 Mask Registers

The mask registers are used to mask the comparison of the data field. Setting these registers to 1s ignore the comparison result of the corresponding data bit.

10.8.2 Debug Control Register

The debug control register is 4-bit wide and is shown in Table 10-10.

Table 10-10. Debug Control Register

3	2	1	0
Single step	INTDIS	DBGRQ	DBGACK

The bit functions of the debug control register are described in Table 10-11.

Table 10-11. Bit Functions of Debug Control Register

Bit	Description	Type
3	Enable single stepping	R/W
2	Disable asynchronous interrupt	R/W
1	Debug request	R/W
0	Debug acknowledgment	Write-only

The single step function can be used without occupying a hardware breakpoint unit. Writing '0' to the DBGRQ and DBGACK registers is necessary before exiting from the debug state.

10.8.3 Debug Status Register

The debug status register is 4-bit wide and is shown in Table 10-12.

Table 10-12. Debug Status Register

4	3	2	1	0
Thumb mode	SYSCOMP	IFEN	DBGRQ	DBGACK

The bit functions of the debug status register are described in Table 10-13.

Table 10-13. Bit Functions of Debug Status Register

Bit	Descriptioin	Type
4	Thumb mode status signal	Read-only
3	System speed instruction completed	Read-only
2	Interrupt enable signal	Read-only
1	Debug request	Read-only
0	Debug acknowledgment	Read-only

10.8.4 Vector Catch Register

The vector catch function is to force the processor core into the debug state when certain specified exceptions had happened. Once entering the debug state, the processor stops at the first instruction of the exception handler. The vector catch register is used to catch the exception. Its format is shown in Table 10-14.

Table 10-14. Vector Catch Register

7	6	5	4	3	2	1	0
FIQ	IRQ	Reserved	Data abort	Pre-fetch abort	SWI	Undefined	Reset

10.8.5 Watchpoint Control Register

The watchpoint control register is 9-bit wide. The bit 3 of this register is used to program the watchpoint unit to monitor the program access or data access.

If bit 3 is set to ‘1’, watchpoint will be used to monitor the data access of the processor core. The fields of this control register for data access monitor are shown in Table 10-15.

Table 10-15. Watchpoint Control Register for Data Access Monitor

8	7	6	5	4	3	2	1	0
ENABLE	RANGE	CHAIN	EXTERN	DnTRANS	1	DMAS[1]	DMAS[0]	DnRW

The bit functions of the watchpoint control register for data access monitor are shown in Table 10-16.

Table 10-16. **Bit Functions of Watchpoint Control Register for Data Access Monitor**

Bit	Name	Description
8	ENABLE	Enable watchpoint This bit cannot be masked.
7	RANGE	RANGEOUT of another watchpoint This bit allows two watchpoints to be coupled to detect two conditions that occurred simultaneously.
6	CHAIN	This bit will be set if another watchpoint meets its conditions. It allows two watchpoints to be coupled to detect two conditions that occurred in the specified sequence.
5	EXTERN	This bit allows an external input signal to be compared as a condition.
4	DnTRANS	This bit is compared with the DnTRANS bit from the processor core to detect whether the data transfer is in the user mode (DnTRANS = 0) or in the privileged mode (DnTRANS = 1).
[2:1]	DMAS	These bits are compared with the DMAS bus from the processor core to detect the size of a data transfer.
0	DnRW	This bit is compared with DnRW bit from the processor core to detect whether the direction of the data transfer is data read (DnRW = 0) or data write (DnRW = 1).

If bit 3 is set to ‘0’, watchpoint can be used to monitor the program access of the processor core. The fields of the control register for the program access monitor are shown in Table 10-17.

Table 10-17. **Watchpoint Control Register for Program Access Monitor**

8	7	6	5	4	3	2	1	0
ENABLE	RANGE	CHAIN	EXTERN	InTRANS	0	X	X	X

The bit functions of the watchpoint control register for program access monitor are shown in Table 10-18.

Table 10-18. **Bit Functions of Watchpoint Control Register for Program Access Monitor**

Bit	Name	Description
8	ENABLE	Enable watchpoint This bit cannot be masked.
7	RANGE	RANGEOUT of another watchpoint This bit allows two watchpoints to be coupled to detect two conditions that occurred simultaneously.
6	CHAIN	This bit will be set if another watchpoint meets its conditions. It allows two watchpoints to be coupled to detect two conditions that occurred in the specified sequence.

Bit	Name	Description
5	EXTERN	This bit allows an external input signal to be compared as a condition.
4	InTRANS	This bit is compared with the InTRANS bit from the processor core to detect whether the program fetch is in the user mode (InTRANS = 0) or in the privileged mode (InTRANS = 1).
[2:0]	Reserved	This is the “Don’t care” bits

The corresponding bits of the mask register should be set to 1’s for the “don’t care” bits.

10.8.6 Watchpoint Address Register and Address Mask Register

The watchpoint address register and the address mask register are both 32-bit wide and can be used to determine the address or address range of the program fetch or data access.

10.8.7 Watchpoint Data Register and Data Mask Register

The watchpoint data register and the data mask register are both 32-bit wide, and can be used to determine the instruction or data pattern for the program fetch or data write.

10.8.8 Debug Communications Register

There are two registers that can be used as the Debug Communication Channel (DCC) between the debug host and the debug target: The DCC control register and the DCC data register. The processor can access these two registers as the CP14 registers by using the instructions in the normal mode, as shown in Table 10-19.

Table 10-19. Instructions for Accessing Debug Communication Register

Register	Instruction	Operation
Debug communication control register	MRC p14, 0, Rd, c0, c0, 0	Read-only
Debug communication data register	MCR p14, 0, Rn, c1, c0, 0	Write
	MRC p14, 0, Rd, c1, c0, 0	Read

The fields of the DCC control register are shown in Table 10-20.

Table 10-20. Debug Communication Control Register

31	30	29	28	27:2	1	0
0	0	1	0	Reserved	W	R

The bit functions of the debug communication control register are described in Table 10-21.

Table 10-21. Bit Functions of Debug Communication Control Register

Bit	Function	Type
[31:28]	ICE module version pattern The version number of the FA626TE processor is 0b0010.	Read-only
[27:2]	Reserved (SBZ)	Read-only
1	Write flag	Read-only
0	Read flag	Read-only

Write flag

When this bit is set to ‘0’, the debug communication data register is ready to be written by the processor. When this bit is set to ‘1’, the previous write data are not read by the debugger. This bit can be set to ‘1’ when the processor writes data to the debug communication data register. This bit will be cleared to ‘0’ if the debugger reads data from the debug communication data register.

Read flag

When this bit is set to ‘0’, the debug communication data register is ready to be written by the debugger. When this bit is set to ‘1’, the previous write data has not been read by the processor. This bit can be set to ‘1’ when the debugger writes data to the debug communication data register. This bit will be cleared to ‘0’ if the processor reads data from the debug communication data register.

Chapter 11

Signal Description

This chapter contains the following sections:

11.1 AHB Signals

Table 11-1. AHB Signals

Signal Name	Direction	Description
IRHBUSREQ	Output	AHB bus request
IRHTRANS[1:0]	Output	Transaction type selection
IRHBURST[2:0]	Output	Burst types selection
IRHSIZE[2:0]	Output	Size type
IRHADDR[31:0]	Output	Address
IRHWDATA[63:0]	Output	AHB write data
IRHLOCK	Output	AHB LOCK indicator
IRHWRITE	Output	AHB write transaction indicator
IRHPROT[3:0]	Output	Attribution selection
IR_CMD_OE	Output	Command output enable
IR_DATA_OE	Output	Data output enable
IR64	Input	32-bit or 64-bit bus width indicator
IRHRDATA[63:0]	Input	Read data
IRHREADY	Input	AHB HREADY signal indicator
IRHGRANT	Input	AHB bus grant signal
IRHRESP[1:0]	Input	AHB response selection
DRHBUSREQ	Output	AHB bus request
DRHTRANS[1:0]	Output	Transaction type selection
DRHBURST[2:0]	Output	Burst type selection
DRHSIZE[2:0]	Output	Transfer size selection
DRHADDR[31:0]	Output	Address
DRHWDATA[63:0]	Output	AHB write data
DRHLOCK	Output	AHB LOCK indicator
DRHWRITE	Output	AHB write transaction indicator
DRHPROT[3:0]	Output	Attribution selection
DR_CMD_OE	Output	Command output enable
DR_DATA_OE	Output	Data output enable
DR64	Input	32-bit or 64-bit bus width indicator
DRHRDATA[63:0]	Input	Read data
DRHREADY	Input	AHB HREADY signal indicator
DRHGRANT	Input	AHB bus grant signal

Signal Name	Direction	Description
DRHRESP[1:0]	Input	AHB response selection
DWHBUSREQ	Output	AHB bus request
DWHTRANS[1:0]	Output	Transaction type selection
DWHBURST[2:0]	Output	Burst type selection
DWHSIZE[2:0]	Output	Transfer size selection
DWHADDR[31:0]	Output	Address
DWHWDATA[63:0]	Output	AHB write data
DWHLOCK	Output	AHB LOCK indicator
DWHWRITE	Output	AHB write transaction indicator
DWHPROT[3:0]	Output	Attribution selection
DW_CMD_OE	Output	Command output enable
DW_DATA_OE	Output	Data output enable
DW64	Input	32-/64-bit bus width indicator
DWHRDATA[63:0]	Input	Read data
DWHREADY	Input	AHB HREADY signal indicator
DWHGRANT	Input	AHB bus grant signal
DWHRESP[1:0]	Input	AHB response selection
PHBUSREQ	Output	AHB bus request
PHTRANS[1:0]	Output	Transaction type selection
PHBURST[2:0]	Output	Burst type selection
PHSIZE[2:0]	Output	Transfer size selection
PHADDR[31:0]	Output	Address
PHWDATA[31:0]	Output	AHB write data
PHLOCK	Output	AHB LOCK indicator
PHWRITE	Output	AHB write transaction indicator
PHPROT[3:0]	Output	Attribution selection
P_CMD_OE	Output	Command output enable
P_DATA_OE	Output	Data output enable
PHRDATA[31:0]	Input	Read data
PHREADY	Input	AHB HREADY signal indicator
PHGRANT	Input	AHB bus grant signal
PHRESP[1:0]	Input	AHB response signal
DMAHTRANS[1:0]	Input	Transaction type selection

Signal Name	Direction	Description
DMAHBURST[2:0]	Input	Burst types FA626TE supports all types.
DMAHSIZE[2:0]	Input	Transfer size selection
DMAHADDR[31:0]	Input	Address
DMAHWDATA[63:0]	Input	AHB write data
DMAHWRITE	Input	AHB write transaction indicator
DMAHPROT[3:0]	Input	Attribution selection
DMAHREADYIN	Input	AHB HREADY signal indicator
DMAHSEL	Input	HSEL signal
DMA64	Input	32-/64-bit bus width indicator
DMAHRDATA[31:0]	Output	Returned read data
DMAHREADYOUT	Output	Slave HREADY signal indicator
DMAHRESP[1:0]	Output	Response signal
DMA_HREADY_OE	Output	Slave HREADY output enable
DMA_DATA_OE	Output	Slave data output enable

11.2 AXI Signals

Table 11-2. AXI Signals

Signal Name	Direction	Description
IARREADY	Input	Address ready used by the slave to indicate the acceptance of the address
IARADDR[31:0]	Output	Initial address of the burst
IARBURST[1:0]	Output	Burst type
IARCACHE[3:0]	Output	Cache type giving additional information about cacheable characteristics
IARLEN[3:0]	Output	Burst length that gives the exact number of transfers
IARLOCK[1:0]	Output	Lock type
IARPROT[2:0]	Output	Protection type
IARSIDEBAND[3:0]	Output	Signals inner cacheable
IARSIZE[2:0]	Output	Burst size
IARVALID	Output	This signal indicates that the address and control are valid.
IDRDATA[63:0]	Input	Read data bus
IDRLAST	Input	This signal indicates the last transfer in a read burst.
IDRRESP[1:0]	Input	Read response indicates the read data is valid

Signal Name	Direction	Description
IDRVALID	Input	This signal indicates that the read data is available.
IDRREADY	Output	Read ready signal indicating that the master can accept the read data and response information
RWARREADY	Input	Address ready, used by the slave to indicate the acceptance of the address
RWARADDR[31:0]	Output	Initial address of the burst
RWARBURST[1:0]	Output	Burst type
RWARCACHE[3:0]	Output	Cache type giving additional information about cacheable characteristics
RWARLEN[3:0]	Output	Burst length that gives the exact number of transfers
RWARLOCK[1:0]	Output	Lock type
RWARPROT[2:0]	Output	Protection type
RWARSIDEBAND[3:0]	Output	Signals inner cacheable
RWARSIZE[2:0]	Output	Burst size
RWARVALID	Output	This signal indicates that the address and control are valid.
RWDRDATA[63:0]	Input	Read data bus
RWDRLAST	Input	This signal indicates the last transfer in a read burst.
RWDRRESP[1:0]	Input	Read response indicates the read data is valid
RWDRVALID	Input	This signal indicates that the read data is available.
RWDRREADY	Output	Read ready signal indicating that the master can accept the read data and response information
RWAWREADY	Input	Address ready, used by the slave to indicate the acceptance of the write address
RWAWADDR[31:0]	Output	Initial address of the write burst
RWAWBURST[1:0]	Output	Write burst type
RWAWCACHE[3:0]	Output	Cache type giving additional information about cacheable characteristics for write accesses
RWAWLEN[3:0]	Output	Write burst length
RWAWLOCK[1:0]	Output	Write lock type
RWAWPROT[2:0]	Output	Write protection type
RWAWSIDEBAND[3:0]	Output	Signals inner cacheable for write accesses
RWAWSIZE[2:0]	Output	Write burst size
RWAWVALID	Output	This signal indicates that the write address and control signals are valid.
RWDWREADY	Input	This signal indicates that the slave is ready to accept the write data.
RWDWDATA[63:0]	Output	Write data bus
RWDWLAST	Output	This signal indicates the last data transfer in a burst.

Signal Name	Direction	Description
RWDWSTRB[7:0]	Output	Write strobes
RWDWVALID	Output	This signal indicates that the valid write data and strobes are available.
RWBRESP[1:0]	Input	Write response
RWBVALID	Input	This signal indicates that a valid write response is available.
RWBREADY	Output	This signal indicates that the core is ready to accept the write response.
PARREADY	Input	Address ready, used by the slave to indicate the acceptance of the address
PARADDR[31:0]	Output	Initial address of the burst
PARBURST[1:0]	Output	Burst type
PARCACHE[3:0]	Output	Cache type giving additional information about cacheable characteristics
PARLEN[3:0]	Output	Burst length that gives the exact number of transfers
PARLOCK[1:0]	Output	Lock type
PARPROT[2:0]	Output	Protection type
PARSIDEBAND[3:0]	Output	Signals inner cacheable
PARSIZE[2:0]	Output	Burst size
PARVALID	Output	This signal indicates that the address and control are valid.
PDRDATA[63:0]	Input	Read data bus
PDRLAST	Input	This signal indicates the last transfer in a read burst.
PDRRESP[1:0]	Input	Read response indicates the read data is valid.
PDRVALID	input	This signal indicates that the read data is available.
PDRREADY	Output	Read ready signal indicating that the master can accept the read data and response information
PAWREADY	Input	Address ready, used by the slave to indicate the acceptance of the write address
PAWADDR[31:0]	Output	Initial address of the write burst
PAWBURST[1:0]	Output	Write burst type
PAWCACHE[3:0]	Output	Cache type giving additional information about cacheable characteristics for write accesses
PAWLEN[3:0]	Output	Write burst length
PAWLOCK[1:0]	Output	Write lock type
PAWPROT[2:0]	Output	Write protection type
PAWSIDEBAND[3:0]	Output	Signals inner cacheable for the write accesses
PAWSIZE[2:0]	Output	Write burst size
PAWVALID	Output	This signal indicates that the write address and control signals are valid.
PDWREADY	Input	This signal indicates that the slave is ready to accept the write data.
PDWDATA[63:0]	Output	Write data bus

Signal Name	Direction	Description
PDWLAST	Output	This signal indicates the last data transfer of a burst.
PDWSTRB[7:0]	Output	Write strobes
PDWVALID	Output	This signal indicates that the valid write data and strobes are available.
PBRESP[1:0]	Input	Write response
PBVALID	Input	This signal indicates that a valid write response is available.
PBREADY	Output	This signal indicates that the core is ready to accept the write response.
DMAARREADY	Input	Address ready used by the slave to indicate the acceptance of the address
DMAARADDR[31:0]	Output	Initial address of the burst
DMAARBURST[1:0]	Output	Burst type
DMAARCACHE[3:0]	Output	Cache type giving additional information about cacheable characteristics
DMAARLEN[3:0]	Output	Burst length that gives the exact number of transfers
DMAARLOCK[1:0]	Output	Lock type
DMAARPROT[2:0]	Output	Protection type
DMAARSIDEBAND[3:0]	Output	Signals inner cacheable
DMAARSIZE[2:0]	Output	Burst size
DMAARVALID	Output	This signal indicates that the address and control are valid.
DMADRDATA[63:0]	Input	Read data bus
DMADRLAST	Input	This signal indicates the last transfer in a read burst.
DMADRRESP[1:0]	Input	Read response indicates that the read data are valid.
DMADRVALID	Input	This signal indicates that the read data are available.
DMADRREADY	Output	Read ready signal indicating that the master can accept the read data and response information
DMAAWREADY	Input	Address ready, used by the slave to indicate the acceptance of the write address
DMAAWADDR[31:0]	Output	Initial address of the write burst
DMAAWBURST[1:0]	Output	Write burst type
DMAAWCACHE[3:0]	Output	Cache type giving additional information about cacheable characteristics for write accesses
DMAAWLEN[3:0]	Output	Write burst length
DMAAWLOCK[1:0]	Output	Write lock type
DMAAWPROT[2:0]	Output	Write protection type
DMAAWSIDEBAND[3:0]	Output	Signals inner cacheable for write accesses
DMAAWSIZE[2:0]	Output	Write burst size
DMAAWVALID	Output	This signal indicates that the write address and control signals are valid.

Signal Name	Direction	Description
DMADWREADY	Input	This signal indicates that the slave is ready to accept write data.
DMADWDATA[63:0]	Output	Write data bus
DMADWLAST	Output	This signal indicates the last data transfer in a burst.
DMADWSTRB[7:0]	Output	Write strobes
DMADWVALID	Output	This signal indicates that valid write data and strobes are available.
DMABRESP[1:0]	Input	Write response
DMABVALID	Input	This signal indicates that a valid write response is available.
DMABREADY	Output	This signal indicates that the core is ready to accept a write response.

11.3 Miscellaneous Signals

Table 11-3. Miscellaneous Signals

Signal Name	Direction	Description
HCLK	Input	Bus clock This pin does not exist in the following IP: FA626TE55EE1001HC0HA
FCLK	Input	CPU clock
edgesyn	Input	Bus clock enable This signal is used to indicate that the CPU cores are at the right rising edge of the bus clock. This pin does not exist in the following IP: FA626TE55EE1001HC0HA
Iedgesyn	Input	Bus clock enable for the instruction port This signal is used to indicate that the CPU cores are at the right rising edge of the bus clock. This pin does not exist in the following IPs: FA626TE54EE0001HC0HA, FA626TE55EE0001HD0AG, FA626TE55EE0001HC0HA, FA626TE55EE0001HE0AG, FA626TE43EE0001HD0AE
RWPedgesyn	Input	Bus clock enable for the RW and P ports This signal is used to indicate that the CPU cores are at the right rising edge of the bus clock. This pin does not exist in the following IPs: FA626TE54EE0001HC0HA, FA626TE55EE0001HD0AG, FA626TE55EE0001HC0HA, FA626TE55EE0001HE0AG, FA626TE43EE0001HD0AE

Signal Name	Direction	Description
DMAedgesyn	Input	Bus clock enable for the DMA port This signal is used to indicate that the CPU cores are at the right rising edge of the bus clock. This pin does not exist in the following IPs: FA626TE54EE0001HC0HA, FA626TE55EE0001HD0AG, FA626TE55EE0001HC0HA, FA626TE55EE0001HE0AG, FA626TE43EE0001HD0AE
RSTN	Input	Global reset signal to CPU
CPUCLKO	Output	Internal CPU clock output which is balanced with the clock leaf of FCLK
HCLKO	Output	Internal HCLK clock output which is balanced with the clock leaf of HCLK This pin does not exist in the following IP: FA626TE55EE1001HC0HA
VINITHI	Input	High V_{ecs} mode Active high
CPUOFFN	Output	This signal indicates that CPU has entered the power-down mode.
IRQN	Input	Interrupt request Active low
FIQN	Input	Fast interrupt request Active low
DMAIRQN	Output	DMA interrupt This pin does not exist in the following IPs: FA626TE54EE0001HC0HA, FA626TE55EE0001HD0AG, FA626TE55EE0001HC0HA, FA626TE55EE0001HE0AG, FA626TE43EE0001HD0AE

11.4 ICE Interface Signals

Table 11-4. ICE Interface Signals

Signal Name	Direction	Description
TCK	Input	ICE clock
IMS	Input	ICE mode selection
IDI	Input	Test data input
IDO	Output	Test data output
IDOE	Output	Test data output enable
nTRST	Input	ICE reset
EXTERN0	Input	Watchpoint0 external condition input
EXTERN1	Input	Watchpoint1 external condition input
EDBGRQ	Input	External debug request
DBGACK	Output	Debug-state acknowledgement

11.5 Scratchpad Interface Signals

Table 11-5. Scratchpad Interface Signals

Signal Name	Direction	Description
DSPCAHOUT[31:0]	Input	Data scratchpad read data
DSPCS	Output	Data scratchpad memory CS Active high
DSPWE0	Output	Byte 0 write enable of the data scratchpad memory Active high
DSPWE1	Output	Byte 1 write enable of the data scratchpad memory Active high
DSPWE2	Output	Byte 2 write enable of the data scratchpad memory Active high
DSPWE3	Output	Byte 3 write enable of the data scratchpad memory Active high
DSPADDR[16:2]	Output	Address of the data scratchpad memory
DSPCAHIN[31:0]	Output	Write data of the external data scratchpad memory
ISPCAHOUT[31:0]	Input	Instruction scratchpad read data
ISPCS	Output	Instruction scratchpad CS Active high

Signal Name	Direction	Description
ISPWE	Output	Write enable of the instruction scratchpad memory Active high
ISPADDR[16:2]	Output	Address of the instruction scratchpad memory
ISPCAHIN[31:0]	Output	Write data of the instruction scratchpad memory

11.6 Test Related Signals

11.6.1 Non At-Speed Test Signals

Table 11-6 Non At-Speed Test Signals

Signal Name	Direction	Description
TCLK	Input	Test clock for TCB
TC_RESET	Input	High-active reset of the TCB To let this IP work in the normal function mode, simply set this signal to high.
TC_SHIFT	Input	Shift enable signal for the shift register inside TCB TC_SHIFT should be set high when shifting in and out the test instructions.
TC_UPDATE	Input	Signal used to update the shift register inside TCB The test instructions shifted into TCB will be functional only by updating TCB.
TC_SI	Input	Scan input of TCB
TC_SO	Output	Scan output of TCB
VC_SHIFT	Input	Shift enable signal for the shift register of the scan chain inside this IP and the wrapper register.
VC_SI[7:0]	Input	Scan inputs of scan chains inside the IP
VC_SO[7:0]	Output	Scan outputs of scan chains inside this IP
WP_CLK	Input	Test clock for wrapper register
WP_SI	Input	Scan input of wrapper register
WP_SO	Output	Scan output of wrapper register
TEST_CLK	Input	Test clock for scan chain and memory BIST inside this IP
TEST_RST	Input	Test reset signal for scan chain and memory BIST inside this IP Active high

Signal Name	Direction	Description
HOLD_L	Output	MBIST hold signal 1'b1: The MBSIT function is used for testing. 1'b 1'b0: The MBIST function is held for diagnosis.
SCAN_OUT	Output	If the MBIST test fails, the failed pattern will be shifted out by this signal.

11.6.2 At-Speed Test Signals

11.6.2.1 At-Speed Test Signals

Table 11-7. At-Speed Test Signals

Signal Name	Direction	Description
EDT_CLOCK	Input	Clock for the EDT logic
EDT_UPDATE	Input	Update signal for the EDT logic
TEST_RST	Input	Test reset signal for the scan chain and memory BIST inside this IP Active high
TEST_CLK1	Input	Test clock for system clock1 (Low speed) and the diagnosis clock for MBIST to shift out the diagnostic information
TEST_CLK2	Input	Test clock for system clock2 (High speed) and the test clock for MBSIT test
TEST_CLK3	Input	Test clock for system clock3 (High speed)
TCLK	Input	Test clock for TCB
TC_RESET	Input	High-active reset of TCB Set this signal to high so that this IP enters the normal function mode
TC_SHIFT	Input	Shift enable signal for the shift register inside TCB TC_SHIFT should be set to high when shifting in and shifting out of the test instructions.
TC_UPDATE	Input	Signal used to update the shift register inside TCB The test instructions shifted into TCB will be functional only by updating TCB.
TC_SI	Input	Scan input of TCB
TC_SO	Output	Scan output of TCB
VC_SHIFT	Input	Shift enable signal for the shift register of the scan chain inside this IP and the wrapper register
VC_SI[3:0]	Input	Scan inputs of scan chains inside the IP
VC_SO[3:0]	Output	Scan outputs of scan chains inside the IP
WP_CLK	Input	Test clock for the wrapper register

Signal Name	Direction	Description
EXTEST_WP_SI0	Input	Scan input of wrapper chain0 for the chip level SCAN/ATPG test
EXTEST_WP_SI1	Input	Scan input of wrapper chain1 for the chip level SCAN/ATPG test
EXTEST_WP_SI2	Input	Scan input of wrapper chain2 for the chip level SCAN/ATPG test
EXTEST_WP_SI3	Input	Scan input of wrapper chain3 for the chip level SCAN/ATPG test
EXTEST_WP_SI4	Input	Scan input of wrapper chain4 for the chip level SCAN/ATPG test
EXTEST_WP_SI5	Input	Scan input of wrapper chain5 for the chip level SCAN/ATPG test
EXTEST_WP_SI6	Input	Scan input of wrapper chain6 for the chip level SCAN/ATPG test
EXTEST_WP_SI7	Input	Scan input of wrapper chain7 for the chip level SCAN/ATPG test
EXTEST_WP_SI8	Input	Scan input of wrapper chain8 for the chip level SCAN/ATPG test
EXTEST_WP_SO0	Output	Scan output of wrapper chain0 for the chip level SCAN/ATPG test
EXTEST_WP_SO1	Output	Scan output of wrapper chain1 for the chip level SCAN/ATPG test
EXTEST_WP_SO2	Output	Scan output of wrapper chain2 for the chip level SCAN/ATPG test
EXTEST_WP_SO3	Output	Scan output of wrapper chain3 for the chip level SCAN/ATPG test
EXTEST_WP_SO4	Output	Scan output of wrapper chain4 for the chip level SCAN/ATPG test
EXTEST_WP_SO5	Output	Scan output of wrapper chain5 for the chip level SCAN/ATPG test
EXTEST_WP_SO6	Output	Scan output of wrapper chain6 for the chip level SCAN/ATPG test
EXTEST_WP_SO7	Output	Scan output of wrapper chain7 for the chip level SCAN/ATPG test
EXTEST_WP_SO8	Output	Scan output of wrapper chain8 for the chip level SCAN/ATPG test
HOLD_L	Input	MBIST hold signal 1’b1: The MBSIT function is used for testing. 1’b 1’b0: The MBIST function is held for diagnosis.
SCAN_OUT	Output	If the MBIST test fails, the failed pattern will be shifted out by this signal.
EDT_BYPASS	Input	Bypass signal for the EDT logic

11.6.2.2 FA626TE55EE1001HC0HA At-Speed Test Signals

Table 11-8. FA626TE55EE1001HC0HA At-Speed Test Signals

Signal Name	Direction	Description
EDT_CLOCK	Input	Clock for the EDT logic
EDT_UPDATE	Input	Update signal for the EDT logic
TEST_RST	Input	Test reset signal for the scan chain and memory BIST inside this IP Active high
TEST_CLK1	Input	Test clock for the system clock1 (Low speed) and the diagnosis clock for MBIST to shift out the diagnostic information
TEST_CLK2	Input	Test clock for the system clock2 (High speed) and the test clock for the MBSIT test
TCLK	Input	Test clock for TCB
TC_RESET	Input	High-active reset of the TCB Set this signal to high so that this IP enters the normal function mode
TC_SHIFT	Input	Shift enable signal for the shift register inside TCB TC_SHIFT should be set to high when shifting in and shifting out the test instructions.
TC_UPDATE	Input	Signal used to update the shift register inside TCB The test instructions shifted into TCB will be functional only by updating TCB.
TC_SI	Input	Scan input of TCB
TC_SO	Output	Scan output of TCB
VC_SHIFT	Input	Shift enable signal for the shift register of the scan chain inside this IP and the wrapper register.
VC_SI[7:0]	Input	Scan inputs of scan chains inside the IP
VC_SO[7:0]	Output	Scan outputs of scan chains inside the IP
WP_CLK	Input	Test clock for the wrapper register
WP_SI	Input	Scan input of the wrapper register
WP_SO	Output	Scan output of the wrapper register
HOLD_L	Input	MBIST hold signal 1’b1: The MBSIT function is used for testing. 1’b 1’b0: The MBIST function is held for diagnosis.
SCAN_OUT	Output	If the MBIST test fails, the failed pattern will be shifted out by this signal.
edt_bypass	Input	Bypass signal for the EDT logic

11.6.2.3 FA626TE55EE0001HE0AG At-Speed Test Signals

Table 11-9. FA626TE55EE0001HE0AG At-Speed Test Signal

Signal Name	Direction	Description
EDT_CLOCK	Input	Clock for the EDT logic
EDT_UPDATE	Input	Update signal for the EDT logic
TEST_RST	Input	Test reset signal for the scan chain and memory BIST inside this IP Active high
TEST_CLK1	Input	Test clock for the system clock1 (Low speed) and the diagnosis clock for MBIST to shift out the diagnostic information
TEST_CLK2	Input	Test clock for the system clock2 (High speed) and the test clock for the MBSIT test
TEST_CLK3	Input	Test clock for the system clock3 (High speed)
TCLK	Input	Test clock for TCB
TC_RESET	Input	High-active reset of the TCB Set this signal to high so that this IP enters the normal function mode
TC_SHIFT	Input	Shift enable signal for the shift register inside TCB TC_SHIFT should be set high when shifting in and out the test instructions.
TC_UPDATE	Input	Signal used to update the shift register inside TCB The test instructions shifted into TCB will be functional only by updating TCB.
TC_SI	Input	Scan input of TCB
TC_SO	Output	Scan output of TCB
VC_SHIFT	Input	Shift enable signal for the shift register of the scan chain inside this IP and the wrapper register.
VC_SI[3:0]	Input	Scan inputs of scan chains inside the IP
VC_SO[3:0]	Output	Scan outputs of scan chains inside the IP
WP_CLK	Input	Test clock for the wrapper register
EXTEST_WP_SI0	Input	Scan input of wrapper chain0 for the chip level SCAN/ATPG test
EXTEST_WP_SI1	Input	Scan input of wrapper chain1 for the chip level SCAN/ATPG test
EXTEST_WP_SI2	Input	Scan input of wrapper chain2 for the chip level SCAN/ATPG test
EXTEST_WP_SI3	Input	Scan input of wrapper chain3 for the chip level SCAN/ATPG test
EXTEST_WP_SI4	Input	Scan input of wrapper chain4 for the chip level SCAN/ATPG test
EXTEST_WP_SI5	Input	Scan input of wrapper chain5 for the chip level SCAN/ATPG test
EXTEST_WP_SI6	Input	Scan input of wrapper chain6 for the chip level SCAN/ATPG test
EXTEST_WP_SI7	Input	Scan input of wrapper chain7 for the chip level SCAN/ATPG test

Signal Name	Direction	Description
EXTEST_WP_SI8	Input	Scan input of wrapper chain8 for the chip level SCAN/ATPG test
EXTEST_WP_SO0	Output	Scan output of wrapper chain0 for the chip level SCAN/ATPG test
EXTEST_WP_SO1	Output	Scan output of wrapper chain1 for the chip level SCAN/ATPG test
EXTEST_WP_SO2	Output	Scan output of wrapper chain2 for the chip level SCAN/ATPG test
EXTEST_WP_SO3	Output	Scan output of wrapper chain3 for the chip level SCAN/ATPG test
EXTEST_WP_SO4	Output	Scan output of wrapper chain4 for the chip level SCAN/ATPG test
EXTEST_WP_SO5	Output	Scan output of wrapper chain5 for the chip level SCAN/ATPG test
EXTEST_WP_SO6	Output	Scan output of wrapper chain6 for the chip level SCAN/ATPG test
EXTEST_WP_SO7	Output	Scan output of wrapper chain7 for the chip level SCAN/ATPG test
EXTEST_WP_SO8	Output	Scan output of wrapper chain8 for the chip level SCAN/ATPG test
HOLD_L	Input	MBIST hold signal 1’b1: The MBSIT function is used for testing. 1’b 1’b0: The MBIST function is held for diagnosis.
SCAN_OUT	Output	If the MBIST test fails, the failed pattern will be shifted out by this signal.
EDT_BYPASS	Input	Bypass signal for the EDT logic

Appendix

This chapter contains the following sections:

Appendix.A Instruction Cycles

A.1. Instruction Execution Cycles

All instructions contain one execution cycle, except for those listed in the following sub-sections.

A.2. Multiply/MAC Execution Cycles

There are six multiply instructions in V5: The MUL, SMULL, UMULL, MLA, SMLAL, and UMLAL multiply instructions. Another five multiply instructions are in the enhanced DSP instructions: The SMULxy, SMLAxy, SMULWy, SMLAWy, and SMLALxy multiply instructions. The multiply and MAC instructions have the same execution cycles. The execution cycles are ranging from one to four cycles depending on the operand sizes, as shown in the following tables:

MUL and MLA

Operand Size	Execution Cycle/Flag-set Cycle
16 x 16	1/2
16 x 32	2/3
32 x 16	1/2
32 x 32	2/3

SMULxy, SMLAxy, SMULWy, and SMLAWy

Operand Size	Execution Cycle/Flag-set Cycle
16 x 16	1/2
32 x 16	1/2

SMULL, SMLAL, UMULL, and UMLAL

Operand Size	RdLo/RdHi/Flag-set Cycle
16 x 16	1/2/3
16 x 32	2/3/4
32 x 16	1/2/3
32 x 32	2/3/4

SMLALxy

Operand Size	RdLo/RdHi/Flag-set Cycle
16 x 16	1/2/3

A.3. Branch Penalty for Missed Prediction

The FA626TE core implements an eight-stage pipeline with the branch prediction mechanism that contains three types of branch instructions. When a branch is predicted correctly, the branch penalty cycle will not be encountered. However, if a branch is predicted incorrectly, there will be certain branch penalty cycles associated with each type of the branch instructions. Each type of the branch instructions has a corresponding branch penalty with incorrect prediction, as shown below. Branches with S-bit set (MOVS PC, LDRS PC) are not predicted by the FA626TE core because these branches cannot be predicted correctly.

Type	Instruction	Description	Penalty Cycle
I	B/BL	Branch or Branch and Link	5
II	MOV/ALU PC	Move to PC	6
III	LDR/LDM PC	Load to PC	7

A.4. LDM/STM

LDM/STM is executed as multiple LDR/STR instructions. Both instructions have the same execution cycles. For example, an LDM with a list of five registers has the same execution cycles as five continuous LDR instructions; that is, one cycle of access time if cache hits.

Appendix.B Pipeline Interlock Cycles

The FA626TE core has eight pipeline stages. The possible dependence and interlock cycles are shown below:

1. ALU operation to ALU/Barrel Shifter operation data dependence: One interlock cycle

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
ADD **R2**, R0, R1	F	I	D	R	S	E	M	W							
SUB R4, R3, **R2**, SHL #1		F	I	D	R	S	S	E	M	W					
…..					F	I	D	R	S	E	M	W			

Figure B-1. ALU-to-ALU/Barrel Shifter Data Dependence (One interlock Cycle)

Please note that the data dependence shown above includes the flag dependence.

1. Load-use data dependence (Cache hit): One interlock cycle

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
LD **R1**, [R0]	F	I	D	R	S	E	M	W							
<instr>		F	I	D	R	S	E	M	W						
ADD R3, R2, **R1**, SHL #1			F	I	D	R	S	S	E	M	W				
…..						F	I	D	R	S	E	M	W		

Figure B-2. Load-use Data Dependence (One Interlock Cycle)

2. Load-use data dependence (Cache hit): Two interlock cycles

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
LD **R1**, [R0]	F	I	D	R	S	E	M	W							
ADD R3, R2, **R1**, SHL #1		F	I	D	R	S	S	S	E	M	W				
…..			F	I	D	R	R	R	S	E	M	W			

Figure B-3. Load-use Data Dependence (Two Interlock Cycles)

3. Load after store (Cache hit): One interlock cycle

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
ST R1, [R0]	F	I	D	R	S	E	M	W							
LD R2, [R0]		F	I	D	R	S	E	M	M	W					
…..			F	I	D	R	S	E	E	M	W				

Figure B-4. Load after Store Interlock

Appendix.C Interrupt Latency

The following figure shows that the interrupt latency is four cycles when there are no data abort and pipeline interlock cycles. Under the worst-case condition, if the interrupt latency happens, the interrupt latency can be calculated as below:

$\mathbf{T_{execution}} + \mathbf{T_{interlock}} + \mathbf{T_{abort}} + \mathbf{T_{fiq}}$ (Four cycles)

where $\mathbf{T_{execution}}$ is the maximum execution cycles (21 cycles for LDM/STM), $\mathbf{T_{interlock}}$ is the possible pipeline interlock cycles caused by data dependence, $\mathbf{T_{abort}}$ is the data abort entry cycles (14 cycles), and $\mathbf{T_{fiq}}$ is the FIQ entry cycles. The interrupt latency does not consider the system operations for cache and TLB managements.

If cache misses are likely to happen, one instruction miss penalty and all possible data miss penalties have to be added to the calculated value based on the formula shown above.

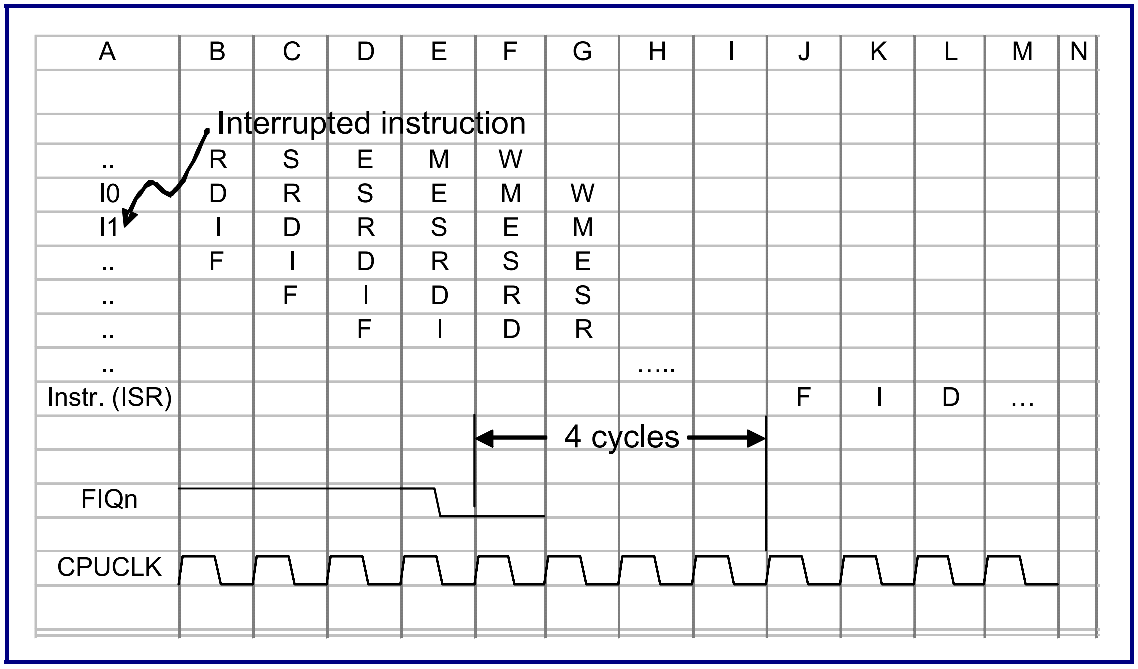

Figure C-1. Minimum Interrupt Latency